|  |
| --- |
|  |
|

**慕风博客**

**MuFeng Blog**

**实现文档**

**Version: [3.0]**

**18301107**王磊、18301108王梦圆、18301113燕怡楠

**2020-12-27**

**All Rights Reserved**

目录

[1 数据库设计 5](#_Toc59996412)

[1.1 引言 5](#_Toc59996413)

[1.1.1 编写目的 5](#_Toc59996414)

[1.1.2 项目来源. 5](#_Toc59996415)

[1.1.3 预期读者和阅读建议 6](#_Toc59996416)

[1.2 数据库命名规则 7](#_Toc59996417)

[1.3 数据库设计说明 9](#_Toc59996418)

[1.3.1 数据库逻辑设计 9](#_Toc59996419)

[1.3.2 数据库物理设计 9](#_Toc59996420)

[1.3.3 数据库分布 10](#_Toc59996421)

[1.3.4 基表设计 10](#_Toc59996422)

[1.3.5 完整性约束 14](#_Toc59996423)

[1.3.6 存储过程设计 15](#_Toc59996424)

[1.3.7 触发器设计 16](#_Toc59996425)

[1.4 词汇表 17](#_Toc59996426)

[2 详细设计 21](#_Toc59996427)

[2.1 登录注册模块 21](#_Toc59996428)

[2.1.1 功能描述 21](#_Toc59996429)

[2.1.2 界面展示 21](#_Toc59996430)

[2.1.3 输入 22](#_Toc59996431)

[2.1.4 输出 22](#_Toc59996432)

[2.1.5 程序逻辑 23](#_Toc59996433)

[2.1.6 限制条件 23](#_Toc59996434)

[2.2 发布博客模块 23](#_Toc59996435)

[2.2.1 功能描述 23](#_Toc59996436)

[2.2.2 界面展示 24](#_Toc59996437)

[2.2.3 输入 24](#_Toc59996438)

[2.2.4 输出 24](#_Toc59996439)

[2.2.5 程序逻辑 24](#_Toc59996440)

[2.2.6 限制条件 24](#_Toc59996441)

[2.3 个人中心模块 25](#_Toc59996442)

[2.3.1 功能描述 25](#_Toc59996443)

[2.3.2 界面展示 27](#_Toc59996444)

[2.3.3 输入 29](#_Toc59996445)

[2.3.4 输出 29](#_Toc59996446)

[2.3.5 程序逻辑 30](#_Toc59996447)

[2.3.6 限制条件 30](#_Toc59996448)

[2.4 搜索模块 30](#_Toc59996449)

[2.4.1 功能描述 30](#_Toc59996450)

[2.4.2 界面展示 31](#_Toc59996451)

[2.4.3 输入 31](#_Toc59996452)

[2.4.4 输出 31](#_Toc59996453)

[2.4.5 程序逻辑 31](#_Toc59996454)

[2.4.6 限制条件 31](#_Toc59996455)

[2.5 标签模块 32](#_Toc59996456)

[2.5.1 功能描述 32](#_Toc59996457)

[2.5.2 界面展示 32](#_Toc59996458)

[2.5.3 输入 32](#_Toc59996459)

[2.5.4 输出 32](#_Toc59996460)

[2.5.5 程序逻辑 33](#_Toc59996461)

[2.5.6 限制条件 33](#_Toc59996462)

[2.6 阅读模块 33](#_Toc59996463)

[2.6.1 功能描述 33](#_Toc59996464)

[2.6.2 界面展示 34](#_Toc59996465)

[2.6.3 输入 34](#_Toc59996466)

[2.6.4 输出 35](#_Toc59996467)

[2.6.5 程序逻辑 35](#_Toc59996468)

[2.6.6 限制条件 35](#_Toc59996469)

[2.7 审核举报模块 35](#_Toc59996470)

[2.7.1 功能描述 35](#_Toc59996471)

[2.7.2 界面展示 36](#_Toc59996472)

[2.7.3 输入 37](#_Toc59996473)

[2.7.4 输出 37](#_Toc59996474)

[2.7.5 程序逻辑 37](#_Toc59996475)

[2.7.6 限制条件 37](#_Toc59996476)

[2.8 活动模块 37](#_Toc59996477)

[2.8.1 功能描述 37](#_Toc59996478)

[2.8.2 界面展示 38](#_Toc59996479)

[2.8.3 输入 38](#_Toc59996480)

[2.8.4 输出 38](#_Toc59996481)

[2.8.5 程序逻辑 38](#_Toc59996482)

[2.8.6 限制条件 39](#_Toc59996483)

[2.9 活动管理模块 39](#_Toc59996484)

[2.9.1 功能描述 39](#_Toc59996485)

[2.9.2 界面展示 39](#_Toc59996486)

[2.9.3 输入 40](#_Toc59996487)

[2.9.4 输出 40](#_Toc59996488)

[2.9.5 程序逻辑 40](#_Toc59996489)

[2.9.6 限制条件 40](#_Toc59996490)

[2.10 视频播放模块 41](#_Toc59996491)

[2.10.1 功能描述 41](#_Toc59996492)

[2.10.2 界面展示 41](#_Toc59996493)

[2.10.3 输入 41](#_Toc59996494)

[2.10.4 输出 41](#_Toc59996495)

[2.10.5 程序逻辑 42](#_Toc59996496)

[2.10.6 限制条件 42](#_Toc59996497)

[3 接口实现 42](#_Toc59996498)

[3.1 index 42](#_Toc59996499)

[3.2 TagCloud 43](#_Toc59996500)

[3.3 info 43](#_Toc59996501)

[3.4 LikeAndCollect 45](#_Toc59996502)

[3.5 Loginbox 46](#_Toc59996503)

[3.6 home 47](#_Toc59996504)

[3.7 blog 49](#_Toc59996505)

[3.8 FollowBtn 51](#_Toc59996506)

[3.9 sort 51](#_Toc59996507)

[3.10 tag 52](#_Toc59996508)

[3.11 admin 53](#_Toc59996509)

[3.12 CommentList 54](#_Toc59996510)

文档修订历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **修订原因** | **版本号** | **作者** | **修订日期** | **备注** |
| **1** | **首次编写** | **1.0** | **燕怡楠** | **2020.12.26** | **无** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 数据库设计

## 引言

本文档为数据库设计说明书。阅读本篇可以帮助你了解web小组的产品“慕风博客”的数据库的设计思路以及实现方式，同时附录相关词汇表和历史数据处理。在阅读本文档时，结合数据库的结构课程所学内容，相信可以帮助你更好的理解与巩固相关知识，为将来参与web开发提前积累经验。

### 编写目的

博客管理系统，是用于阅读博客、撰写博客、后台管理博客的大型网站，集使用和管理于一体，除了基础功能外，还包括标签、活动、个人积分系统、信誉积分系统等附加功能，用户可通过浏览器登陆网站进行相应操作。本说明书给出该系统的使用说明。

编写目的：

 引导用户使用常用功能

 查询手册，便于用户查询相关操作

### 项目来源.

博客管理系统功能丰富、全面，可满足大部分用户对于博客网站的使用需求。

对于用户来说，除了可以完成发博客、看博客、关键词检索、点赞、评论、关注、收藏、举报等一系列基本操作之外，还加入了个人积分系统和信誉积分系统。个人积分系统即每个人都有一个个人积分，个人积分可用于查看收费博客。用户可通过发布博客、参加活动等方式获取个人积分。信誉积分系统是为了保障网站语言环境及风气设立，防止恶行博客发布以及恶意举报的事件发生，用户举报失败或者发布的博客被举报都会被扣除信誉积分，用户举报成功时可以获得信誉积分。信誉积分初始值为5，当低于5分时，根据信誉积分不同会进行看收费博客所需个人积分增多、对于评论、发博、看博客进行限制等一系列惩罚。当信誉积分为0时，会进行封号。

对于管理员来说，可以完成举报信息审核和活动管理两项功能。举报信息管理中，可以根据举报理由进行审核成功或者审核失败的操作。在活动管理中，可以查看所有活动或进行新活动的发布。

在系统设计过程中，开发团队充分考虑到用户体验感，在界面美化、消息反馈机制等方面做出了做出了努力，整体界面风格统一、简洁美观、内容清晰。当用户进行操作时，通过消息提示框的方式进行反馈。同时，当举报成功或者举报失败时，用户都会收到消息提示。

### 预期读者和阅读建议

本文档预期读者：

● 开发人员;

● 项目经理;

● 测试人员;

● 文档编写人员。

阅读建议：首先阅读引言第一部分大致了解本文档的主要内容，随后浏览词汇表对可能出现的词汇提前知悉，然后开始正文部分，最后可以结合“慕风博客”的数据库设计去对比历史数据处理的正确与否。

## 数据库命名规则

1. 规范：遵循行业规范

当有相关国家/行业强制性数据结构标准规范存在时，用于存储某业务数据的业务表在表名命名上原则上应该遵从标准规定，其表中相关字段的中文名称（即数据项名称）若标准规范上有规定的应遵循规定。此外，若标准规范上对数据项的类型、长度有规定的，原则上也应当遵循或保证能直接兼容保存和访问。

2. 规范：字母全部大写原则

所有数据库对象命名字母全部大写。Oracle对大小写不敏感，但是有些数据库对大小写敏感，统一大写有助于在多个数据库间移植。

3. 规范：字符范围原则

只能使用英文字母、下划线、数字进行命名，首位字符必须是英文字母。

4. 规范：分段命名原则

命名中多个单词间采用下划线分隔，以便阅读同时方便某些工具对数据库对象的映射。如XXX\_XXX\_XXX，但不限于三段式。

5. 规范：勿用保留词

数据库对象命名不能直接使用数据库保留关键字，但分段中可以使用。如USER不能用于表名、列名等，但是USER\_NAME可以用于列名，USER\_INFO也可以用于表名。详细保留关键字请参见最后第6.1节，保留字。

6. 规范：简单命名原则

命名尽可能简单，避免太长的命名，尽量使用缩写形式，但是缩写也要能够表达命名的含义。数据库对象命名总长度不得超过30字节，以免超过数据库命名长度限制（Oracle有30的限制，Mysql为64，SQL SERVER也是64）。建议每个单词分段长度不要超过6位。

7. 建议：富有含义原则

数据库对象命名通常用能表示其内容或者含义的英文单词或其缩写表示也可用其中文名称各字词的拼音首写字母或者拼音简写方式表示。数字应尽量避免使用。

此外在公安行业，对于业务表上表示业务属性的字段名（即字段英文名）的命名，业内普遍默认的规范通常是以其中文名称的每个汉字拼音首字母组成。考虑行业习惯和通常思路建议用：建议用于表示用户业务应用属性的数据项字段名采用中文拼音首字母命名，对于其它纯粹用于应用系统内部使用的则尽量使用英文单词进行命名。另外，当按中文名称拼音首字母组合出来后出现与其它字段名重名时，则将最后命名的这个数据项的最后一个汉字用其完整拼音字母代替。

8. 建议：同义性原则

对于同一含义尽量使用相同的单词命名，不管使用英文单词、英文缩写还是拼音首字母，以免引起误解。如TELEPNHOE的A表中表示固定电话号码，在B表中就不应该用于表示移动电话号码。尽量避免同一单词表示多种含义的情况。

9. 建议：命名方式一致原则

在一个系统、一个项目中尽量采用一致的命名方式，都采用英文单词或者拼音首字母。尤其要避免在一个对象命名中同时采用英文单词和拼音首字母。如确实需要在一个项目中采用两种命名方式，考虑系统功能设计相关表（开发）使用英文单词命名，业务相关的表（实施）使用拼音首字母。

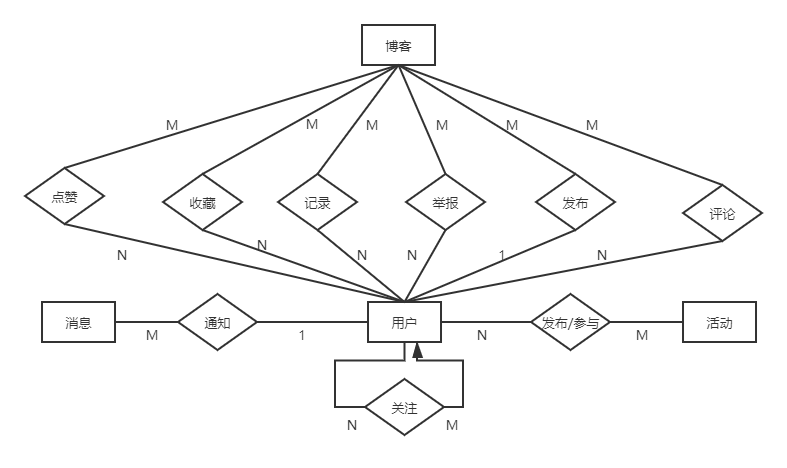
10. 建议：扩展性原则

各系统或者项目在遵循本规范的基础上可以根据需要制定更明确的规范细则，以满足项目管理需要。如对模块进行统一命名，然后用于表名的前缀。建议每个系统在启动开发时建立数据字典，管理命名中使用的英文单词、英文单词缩写、拼音首字母缩写等，对用于命名的单词进行统一管理。

## 数据库设计说明

“慕风博客”主要采用了架设在阿里云服务器上的Mysql数据库。

### 数据库逻辑设计

本系统数据库按照面向对象的思想，设计对应实体类，由实体类生成对应的数据库表，数据精心整理表中的关系，反映了对象间的关系。其中逻辑设计的E-R图如下：

### 数据库物理设计

数据库中主要存储用户的个人信息以及博客等内容，其中用户的个人信息中也包括了点赞、收藏、浏览历史、我的回复等用户方便用户进行博客交流的数据。

### 数据库分布

如上文中提到，本系统中包含一个数据库：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据库编号 | 数据库管理系统名称 | 数据库管理系统版本号 | 数据库英文名称 | 数据库中文名称 | 数据库安装物理位置 |
| 1 | 阿里云云端数据库Mysql | 1.0 | Blog | 慕风博客 | 阿里云平台 |

### 基表设计

数据库中永久存储的表，并且基表就是实际存在的表，它是实际存储数据的逻辑表示。基表由[视图](https://baike.baidu.com/item/%E8%A7%86%E5%9B%BE)、[游标](https://baike.baidu.com/item/%E6%B8%B8%E6%A0%87)、[SQL](https://baike.baidu.com/item/SQL) 语句和[存储过程](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B)引用。后文中提到的子表多由基表数据以及特定数据按照不同的条件衍生而来。

说明：单个表的description 中 ，标有PK的所有属性共同构成主键。

1. 用户Blog\_user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| attributes | Internal name | datatype | Null or not | Content | Description |
| User\_id | user\_id | varchar(10) | Not null | 用户登录ID | PK |
| User\_name | name | varchar(20) | Not null | 用户名 |  |
| User\_password | password | vachar(20) | Not null | 用户登录密码 |  |
| User\_email | email | vachar(50) | Not null | 用户绑定邮箱 |  |
| User\_credit | credit | Int(11) | Not null | 用户积分 | 个人积分可以用来解锁精华博客 |
| User\_role | role | Int(11) | Not null | 用户角色 | 0表示普通用户；  1表示管理员用户。 |
| User\_state | state | Int(11) | Not null | 当前账号状态 | 0表示已经被封号  1表示用户登录状态正常 |
| User\_reputation | Reputation | Int(11) | Not null | 当前状态信誉积分 | 信誉积分由高到低分别设置5，4，3，2，1，0；六个等级，不同积分的用户具有不同的权限。 |

1. 博客表Blog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| attributes | Internal name | datatype | Null or not | Content | Description |
| Blog\_id | blog\_id | Char(10) | Not null | 已发博客ID | PK |
| Author\_id | user\_id | Char(10) | Not null | 发布作者ID | FK |
| Publish\_Time | publish\_time | datetime | Not null | 博文发布日期 |  |
| Blog\_Name | title | Char(200) | Not null | 博客名 |  |
| Blog\_label | label | varchar(20) | Not null | 文件标签 |  |
| Blog\_content | content | Char(5000) | Not null | 博客内容 |  |
| Cnt\_of\_approval | approval\_number | Int(11) | Not null | 文章点赞数 |  |
| Cnt\_of\_browse | browse\_number | Int(11) | Not null | 文章浏览量 |  |
| Need credit | need\_credit | Int(11) | Not null | 查看所需积分 |  |
| state | state | Int(11) | Not null | 1表示文章状态正常，0表示文章已被举报/下架 |  |
| Activity | activity | Int(11) | Not null | 所选数字代表所参与活动的数字编号 |  |

1. 评论Comment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| Comment\_id | comment\_id | Int(11) | Not bull | 评论ID | PK |
| user\_id | user\_id | Char(10) | Not null | 用户ID | FK |
| Blog\_id | blog\_id | Char(10) | Not null | 博客ID | FK |
| Comment | content | Char(100) | Not null | 评论内容 |  |
| Comment\_time | time | datetime | Not null | 评论时间 |  |

1. 历史浏览记录 History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| user\_id | user\_id | varchar(10) | Not null | 用户ID | PK |
| Blog\_id | B\_id | Char(10) | Not null | 博客ID | PK |
| browsing\_time | time | Date() | Not null | 浏览时间 | PK |

1. 收藏 Favorites

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| user\_id | user\_id | Char(20) | Not null | 用户ID | PK,FK |
| Blog\_id | blog\_id | Char(20) | Not null | 博客ID | PK,FK |
| Time | Time | Date() | Not null | 收藏时间 |  |
| Name | name | varchar(20) | Not null | 收藏名 |  |

1. 关注 follower

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| user\_id | user\_id | Char(20) | Not null | 被关注用户ID | PK ,FK |
| follower\_id | follower\_id | Char(20) | Not null | 关注者ID | PK,FK |
| Follow\_time | time | datetime | Not null | 关注时间 |  |

1. 举报report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| user\_id | user\_id | varchar(20) | Not null | 用户ID | PK,FK |
| Report\_time | time | datetime | Not null | 举报时间 | PK |
| Reported\_bolg\_id | blog\_id | Int(11) | Not null | 被举报文章ID | PK,FK |
| Report\_reason | reason | varchar(500) | Not null | 举报理由 |  |
| Report\_state | state | Int(11) | Not null | 举报状态 | 0表示未处理  1表示举报成功  -1表示举报失败 |

1. 系统消息Blog\_message

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| Message\_id | message\_id | Int(11) | Not null | 消息ID | PK |
| User\_id | user\_id | varchar(20) | Not null | 用户ID | FK |
| Message\_content | content | varchar(5000) | Not null | 消息内容 |  |
| Message\_type | type | Int(11) | Not null | 消息类型 | 包括点赞，收藏，评论，举报，账号冻结 |

1. 系统消息Activity

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| Activity\_id | Activity\_id | Int(11) | Not null | 活动ID | PK |
| User\_id | user\_id | varchar(20) | Not null | 用户ID | FK |
| Activity\_name | name | varchar(20) | Not null | 活动名称 |  |
| Start\_time | Start\_time | datetime | Not null | 开始时间 |  |
| end\_time | End\_time | datetime | Not null | 终止时间 |  |
| credit | credit | Int(11) | Not null | 参与活动可获得积分 |  |
| Description | Description | Varchar(50) | Not null | 活动描述 |  |

1. 点赞表 Approval

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Internal name | datatype | Null or not | Content | Description |
| user\_id | user\_id | Char(20) | Not null | 用户ID | PK,FK |
| Blog\_id | blog\_id | Char(20) | Not null | 博客ID | PK,FK |
| Time | Time | Date() | Not null | 点赞时间 |  |

### 完整性约束

**数据的完整性**

约束是用来确保数据的准确性和一致性。数据的完整性就是对数据的准确性和一致性的一种保证。

数据完整性(Data Integrity)是指数据的精确(Accuracy)和可靠性(Reliability)。

分为以下四类：

1) 实体完整性：规定表的每一行在表中是惟一的实体。

2) 域完整性：是指表中的列必须满足某种特定的数据类型约束，其中约束又包括取值范围、精度等规定。

3) 参照完整性：是指两个表的[主关键字](https://baike.baidu.com/item/%E4%B8%BB%E5%85%B3%E9%94%AE%E5%AD%97)和[外关键字](https://baike.baidu.com/item/%E5%A4%96%E5%85%B3%E9%94%AE%E5%AD%97/1239625)的数据应一致，保证了表之间的数据的一致性，防止了数据丢失或无意义的数据在数据库中扩散。

4) 用户定义的完整性：不同的[关系数据库系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F)根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性即是针对某个特定关系数据库的约束条件，它反映某一具体应用必须满足的语义要求。

**完整性约束的类型**

可分为三种类型：与表有关的约束、域(Domain)约束、断言(Assertion)

1) 与表有关的约束：是表中定义的一种约束。可在列定义时定义该约束，此时称为列约束，也可以在表定义时定义约束，此时称为表约束。

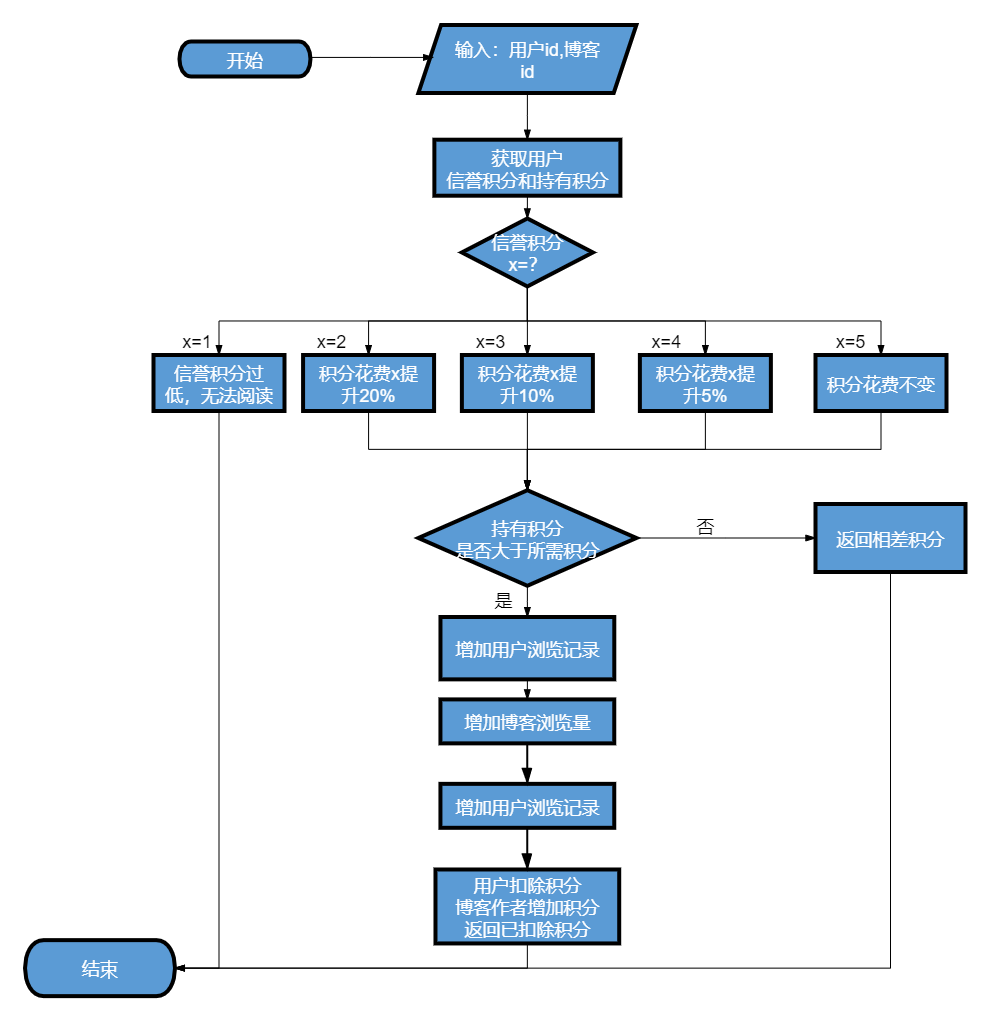
2) 域(Domain)约束：在域定义中被定义的一种约束，它与在特定域中定义的任何列都有关系。

3) 断言(Assertion)：在断言定义时定义的一种约束，它可以与一个或多个表进行关联。

### 存储过程设计

存储过程（Stored Procedure）是在大型[数据库系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F/215176)中，一组为了完成特定功能的SQL 语句集，它存储在数据库中，一次[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343)后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。在数据量特别庞大的情况下利用存储过程能达到倍速的效率提升。

存储过程为阅读收费博客这一过程，包括获取用户的信誉积分和持有积分，根据用户的信誉积分和博客所需积分计算出该用户看博客所需的积分，和用户持有积分进行比较，判断用户是否可以浏览该篇博客。若可以，修改浏览记录、修改该用户的积分和博主的积分、增加博客浏览量。具体流程图如下：



### 触发器设计

触发器为管理员审核举报。系统设定若多人举报同一篇博客，只给第一个举报该博客成功的用户加分，其余用户既不加分、也不减分。若该博客已经被举报成功，触发器直接结束。若该篇博客被没有举报成功过，根据此次举报是否成功进行相应操作。若举报成功，修改被举报者的状态、举报者的状态和博客状态，在被举报者没有被封号的情况下，扣除信誉积分，并根据信誉积分判断是否需要封号。在举报信誉积分不足5的情况下，增加信誉积分。对该篇博客进行封锁，禁止查看。若举报失败，修改举报者的状态。在举报者没有被封号的情况下，扣除信誉积分。具体流程图如下：



## 词汇表

本部分为补充内容，用于帮助读者更好阅读“慕风博客”文档中可能出现的不常见专业术语内容。以下根据首字母进行排序，读者可以根据需求进行快速查找：

1. Access method（访问方法）：此步骤包括从文件中存储和检索记录。

2. Alias（别名）：某属性的另一个名字。在SQL中，可以用别名替换表名。

3. Alternate keys（备用键，ER/关系模型）：在实体/表中没有被选为主健的候选键。

4. Anomalies（异常）参见更新异常（update anomalies）

5. Application design（应用程序设计）：数据库应用程序生命周期的一个阶段，包括设计用户界面以及使用和处理数据库的应用程序。

6. Attribute（属性）（关系模型）：属性是关系中命名的列。

7. Attribute（属性）（ER模型）：实体或关系中的一个性质。

8. Attribute inheritance（属性继承）：子类成员可以拥有其特有的属性，并且继承那些与超类有关的属性的过程。

9. Base table（基本表）：一个命名的表，其记录物理的存储在数据库中。

10. Binary relationship（二元关系）：一个ER术语，用于描述两个实体间的关系。例如，panch Has Staff。

11. Bottom-up approach（自底向上方法）：用于数据库设计，一种设计方法学，他从标识每个设计组建开始，然后将这些组件聚合成一个大的单元。在数据库设计中，可以从表示属性开始底层设计，然后将这些属性组合在一起构成代表实体和关系的表。

12. Business rules（业务规则）：由用户或数据库的管理者指定的附加规则。

13. Candidate key（候选键，ER关系模型）：仅包含唯一标识实体所必须得最小数量的属性/列的超键。

14. Cardinality（基数）：描述每个参与实体的可能的关系数目。

15. Centralized approach（集中化方法,用于数据库设计）：将每个用户试图的需求合并成新数据库应用程序的一个需求集合

16. Chasm trap（深坑陷阱）：假设实体间存在一根，但某些实体间不存在通路。

17. Client（客户端）：向一个或多个服务器请求服务的软件应用程序。

18. Clustering field（群集字段）：记录总的任何用于群集（集合）航记录的非键字段，这些行在这个字段上有相同的值。

19. Clustering index（群集索引）：在文件的群集字段上定义的索引。一个文件最多有一个主索引或一个群集索引。

20. Column（列）：参加属性（attribute）。

21. Complex relationship（复杂关系）：度数大于2的关系。

22. Composite attribute（复合属性）：由多个简单组件组成的属性。

23. Composite key（复合键）：包含多个列的主健。

24. Concurrency control（并发控制）：在多用户环境下同时执行多个十五并保证数据完整性的一个DBMS服务。

25. Constraint（约束）：数据库不允许包含错误数据的一致性规则。

26. Data conversion and loading（数据转换和加载）：数据库应用生命周期重的一个阶段，包括转换现有数据到新数据库中以及酱下耨应用程序转换到新的数据库上运行。

27. Data dictionary（数据字典）：参见系统目录（system catalog）。

28. Data independence（数据独立性）：使用数据的应用程序的数据描述部分。这意味着，如果将新的数据结构添加到数据库中，或者数据库中现有的结构被修改了，那么使用此数据库的就会受到影响，除非应用程序不直接依赖于被修改的部分。

29. Data model（数据模型）：描述数据、数据间关系以及数据的约束的概念的一个集成的集合。

30. Data redundancy（数据冗余）：参见冗余数据（redundant data）。

31. Data security（数据安全）：包括对数据库对象（如表和视图）的访问和使用以及用户可以在这些对象上实施的操作。

32. Database（数据库）：是逻辑上相关的数据（以及这些数据的描述）的一个共享的集合，用于解决公司对信息的需求。

33. Database design（数据库设计）：数据库应用生命周期中的一个阶段，包括创建一个支持公司的操作和目标的数据库的设计。

34. Database integrity（数据库完整性）：指存储数据的正确定和一致性。完整性通常用约束来表达。

35. Database Management System，DBMS（数据库管理系统）：一个能够让用户定义、创建和维护数据库并控制对数据库的访问的软件系统。

36. Database planning（数据库规划）：能尽可能有效的实现数据库应用的各阶段的管理活动。

37. Database server（数据库服务器）：同服务器。

38. DBMS engine（DBMS引擎）：同服务器。

39. DBMS selection（DBMS选择）：数据库应用生命周期中的一个阶段，包括选择一个合适的DBMS来支持数据库应用。

40. Degree of a relationship（关系的度）：一个关系中参与的实体的个数。

41. Denormalization（反规范化）：形式上，这个术语指的是对基本表结构的修改，这样新的表比原始的表的规范化程度要低。但也可以用此属于更宽泛地形容将两个表和并成一个新表的情形，而这个新表与原来的表具有相同的范式，但比原表包含更多的空值。

42. Derived attribute（派生属性）：表示其值可以从一个相关属性和属性集的值派生得到的属性，这个属性在实体中不是必须的。

43. Design methodology（设计方法学）：一种结构化的方法，它使用过程、工具和文档来支持和简化设计过程。

44. Disjoint constraint（无连接约束）：描述子类的成员间的关系，并指明超类某个成员是否有可能成为一个或多个子类的成员。

45. Domain（域）：一个或多个属性的取值范围。

46. Entity（实体）：具有相同性质的对象的集合，它是由用户或公司标识并可独立存在的。

47. Entity integrity（实体完整性）：在一个基本表中，主健列的值不能为空。

48. Entity occurrence（实体出现）：实体中的一个唯一可标识的对象。

49. Entity-Relationship model（实体关系模型）：公司的实体、属性和关系的详细逻辑表示。

50. Fact-finding（事实发现）：使用诸如面谈和提问等技术收集关于系统的事实、需求和性能的形式化过程。

51. Fan trap（扇形陷阱）：但从第三个实体扇出的两个实体有1:\*关系时出现扇形陷阱，但这两个实体在他们之间应该有直接关系以提供必要的信息。

52. Field（字段）：同元组（Tuple）。

53. File（文件）：存储在副主存储器中的相关记录的一个命名集合。

54. File-based system（基于文件的系统）：一个文件集合，用来管理（创建、插入、删除、更新和检索）一个或多个文件中的数据，并产生基于这些文件中的数据的应用（通常是报表）。

55. File organization（文件组织）：当文件存储在磁盘上时，对文件中的记录的安排方式。

56. First normal form（1NF，第一范式）：表中的每个列的交叉处以及记录包含切进包含一个值的表。

57. Foreign key（外健）：一个表中的一个列或者多个列的集合，这些列匹配某些其他（也可能是同一个）表中的候选键。

58. 4GL, Fourth-Generation Language（第四代语言）：一种非过程化语言，比如SQL，他只需要用户定义必须完成什么操作，4GL负责将所进行的操作翻译成如何实现这些操作。

59. Full functional dependency（完全函数依赖）：一个列在功能上依赖于复合主健，但不依赖于主健的任何一个子集的条件。

60. Functional dependency（函数依赖）：描述表中列之间的关系。

61. Generalization（泛化）：通过标识实体间的公共特征使实体间差别最小化的过程。

62. Generalization hierarchy（泛化层次结构）：同类型层次（type hierarchy）。

63. Global data model（全局数据模型）：代表整个公司（和被模型化的公司的一部分）的数据模型。

64. Implementation（实现）：数据库应用生命周期中的一个阶段，包括数据库和应用程序设计的物理实现。

65. Index（索引）：一种允许DBMS将特定的记录更快的放置到文件中，从而加快对用户查询的响应的数据结构。

66. Infomation system（信息系统）：能够在整个公司范围内收集、管理、控制和分发数据/信息的资源。

67. Inheritance（继承）：参见属性继承（attribute inheritance）。

68. Integrity constaints（完整性约束）：防止出现数据库中的数据不一致的约束。

69. IS-A hierarchy（IS－A层次结构）：同类型层次结构（type hierarchy）。

70. Local logical data model（局部逻辑数据模型）：代表特定用户视图或用户视图的组合的数据模型。

71. Logical database design（逻辑数据库设计）：基于特定的数据模型构建公司的数据的模型的过程，但不依赖于特定的DBMS以及其他的物理条件。

72. Meta-data（元数据）：关于数据的数据，参见系统目录（system catalog）。

73. Mision objective（使命目标）：标识数据库必须支持的特定任务。

74. Mission statement（使命语句）：定义数据库应用程序的主要目标。

75. Multiplicity（多样性）：定义与某个相关实体的一次出现有关的实体的出现数目。

76. Multi-valued attribute（多值属性）：为一个实体的出现保存多个值的属性。

77. Nonkey attribute/column（非键属性/列）：不是键的一部分的属性/列。

78. Normal forms（范式）：规范化过程的一个阶段。前三个范式分别为第一范式（1NF）、第二范式（2NF）、第三范式（3NF）。

79. Normalization（规范化）：一种产生带有需要的特性的技术，这种特性能支持用户和公司的需求。

80. Null（空值）：表示当前不知道或对于这条记录来说不可使用的一个列的值。

# 详细设计

## 登录注册模块

### 功能描述

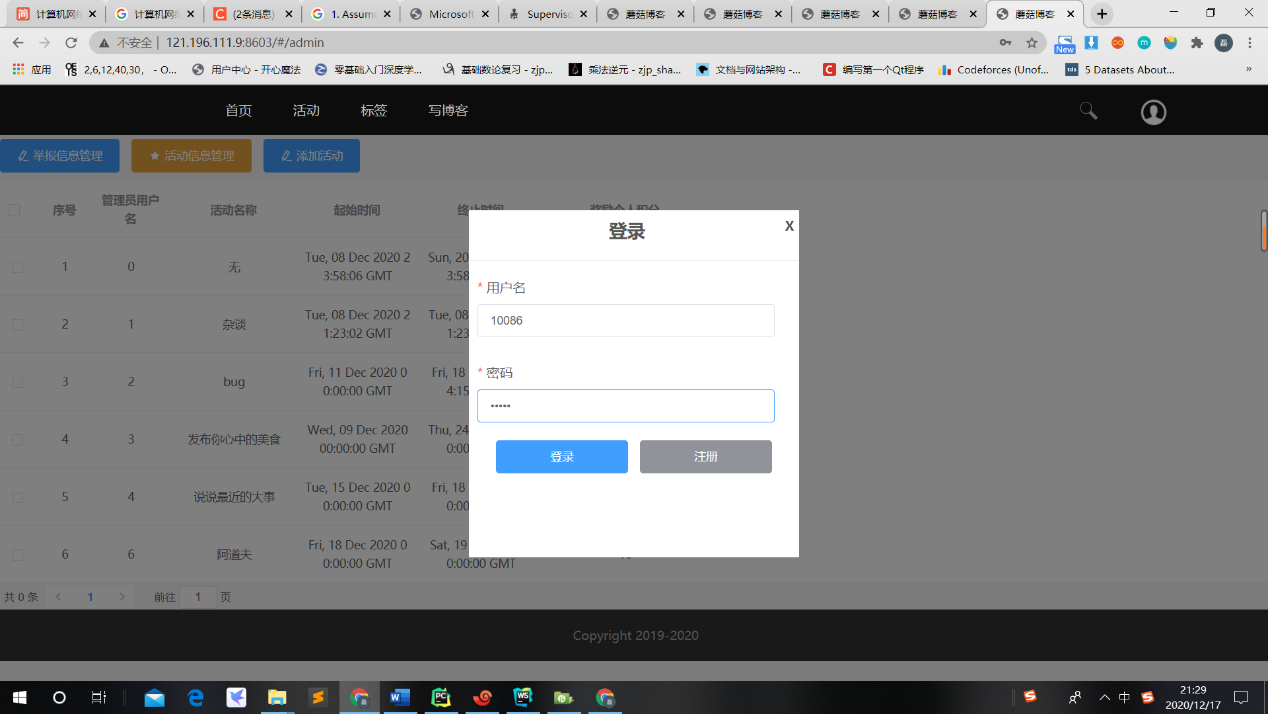
慕风博客网页向阿里云服务器发送登录注册请求，通过阿里云服务器进行网络请求并发处理以及数据库的增删改查，性能较好。

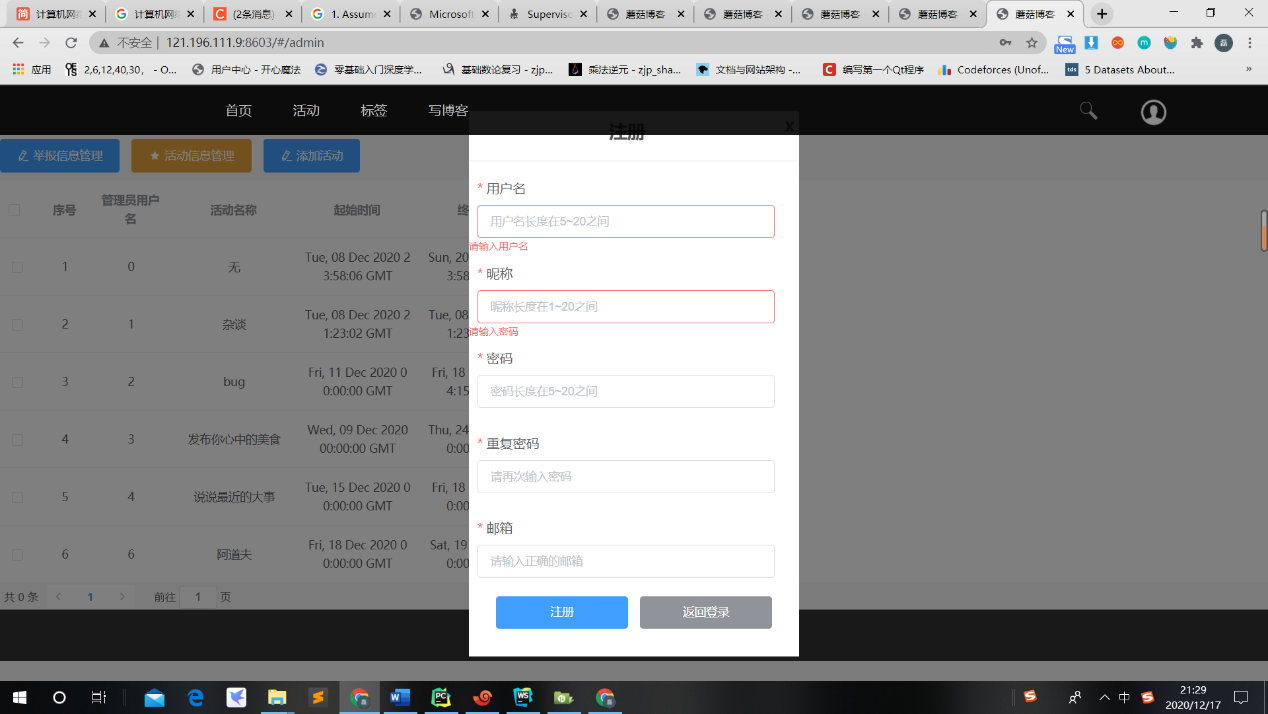
注册：新用户通过输入用户名，密码和邮箱进行注册。

登录：用户通过输入用户名密码进行登录

忘记密码：点击忘记密码，向用户注册时输入的邮箱发送邮件，用户通过邮件可以重置密码。

### 界面展示





### 输入

注册：用户名，密码， 邮箱

登录：用户名，密码

忘记密码：用户名，密码

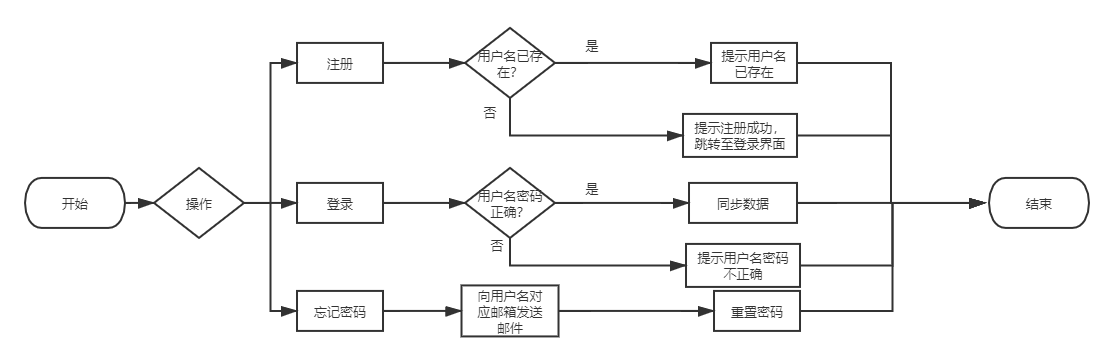
### 输出

注册：用户注册信息，用户名已存在或成功登录

登录：用户登录信息，登录成功，或者用户名密码错误

忘记密码：重置密码邮件

### 程序逻辑



### 限制条件

用户在联网情况下才能与阿里云服务器进行交互，实现登录注册忘记密码功能。

## 发布博客模块

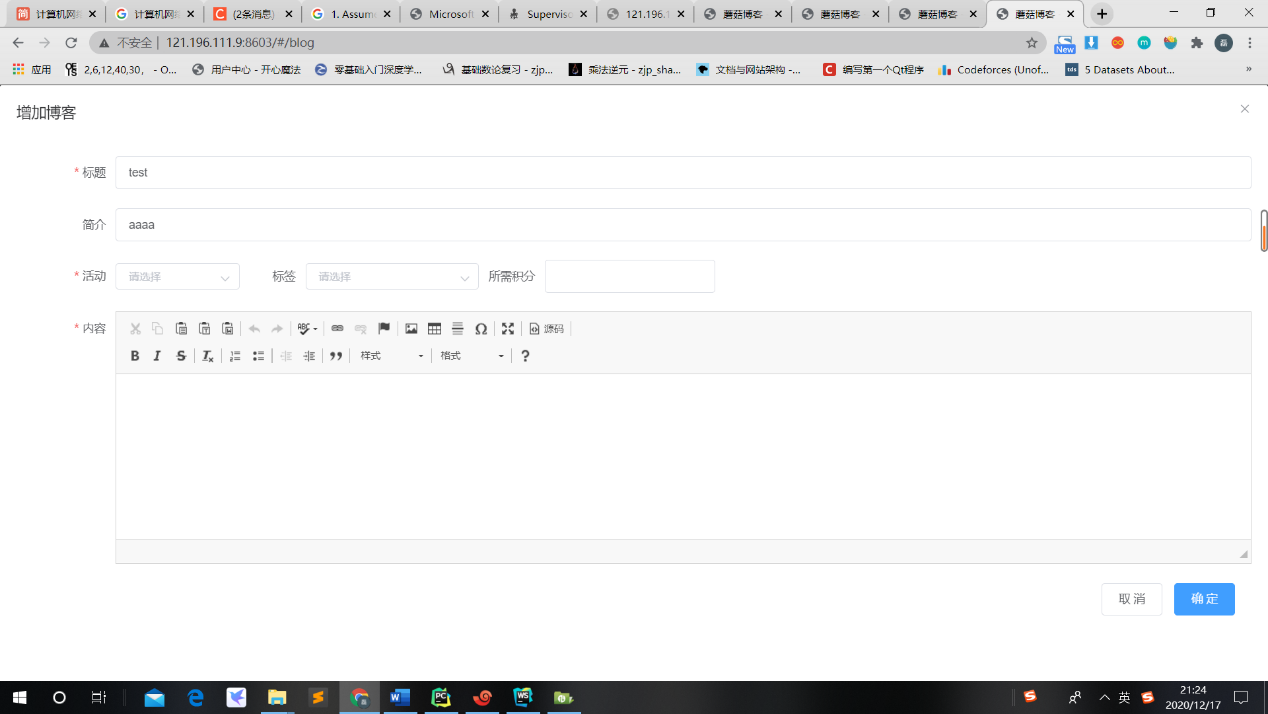
### 功能描述

用户登录后，点击首页的“写博客”选项，即可进入到写博客的功能界面。

进入后即可编辑文章标题以及文章简介。

用户可以选择给自己的文章添加正在进行的活动。

### 界面展示



### 输入

用户在写博客界面对应输入框输入相应内容。

### 输出

提示用户发布博客成功。

### 程序逻辑

发布博客

开始

结束

### 限制条件

需要拟定标题以及填写正文内容。

## 个人中心模块

### 功能描述

当用户登陆成功以后，点击右上角的用户头像将会显示个人中心按钮，点击即可进入个人中心界面。

在个人中心界面中，用户可以查看自己的信息，包括：昵称、信誉积分、积分、邮箱以及简介等，其中积分以及信誉积分不可擅自更改，因此将这两栏输入框设置为了不可更改。

用户在个人中心界面点击最上方加号按钮，可以自定义个人头像。

用户还可以通过选择本地图片上传成为头像。拖动图片到截图所示窗口位置或者点击窗口打开文件选择器，即可选择图片。

用户可以自由调整图片裁剪位置，进行图片缩放，右侧还可以预览头像效果。点击上一步按钮用户可以返回上一步操作。点击保存，用户即可看到自己个人中心的头像已经变为之前选择的图片。

用户浏览网页时，可以对博客进行评论、点赞、收藏、举报、关注等操作，这些行为都将会在个人中心的相关栏目中进行记录提示。

(1) 我的评论：

用户在浏览博客并进行评论以后，将会将这一消息显示在我的评论中。用户可以通过这一板块查看我评论过的文章，并通过点击相关消息跳转到我评论的这篇文章的详情界面。

(2) 我的回复：

用户发布博客以后可以通过这一板块查看在我发布的博客下进行了评论的相关消息，同时将会显示评论人的用户名。通过点击相关消息将会跳转到这个评论所在的博客界面。

(3) 我的点赞：

用户在浏览博客是可以进行点赞，点赞成功后将会在我的点赞模块发布有关消息。用户可以在这一模块查看我赞过的所有博客，并通过点击实现跳转到目标博客的工作。

(4) 我的反馈：

用户在浏览博客时如果认为博客又不合法或者又不合理的地方，可以通过举报并填写举报理由来对博客进行举报。管理员审核完成后将会提示举报者举报是否成功，并在我的反馈板块中进行显示。

(5) 我的收藏：

用户在浏览博客后可以选择对相关博客进行收藏。收藏时可以对博客进行再命名，以方便用户在想再次浏览博客的时候更快速简介的跳转。点击相关消息将会跳转到收藏的博客所在界面。

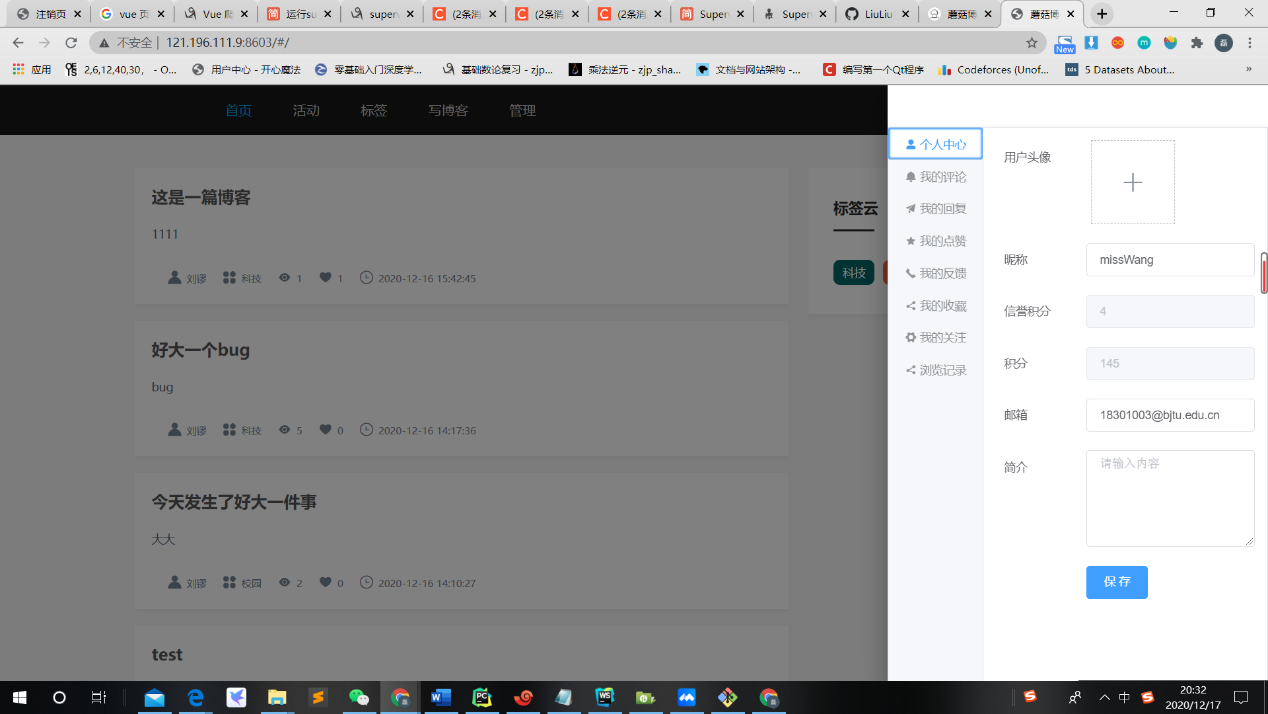
(6) 我的关注：

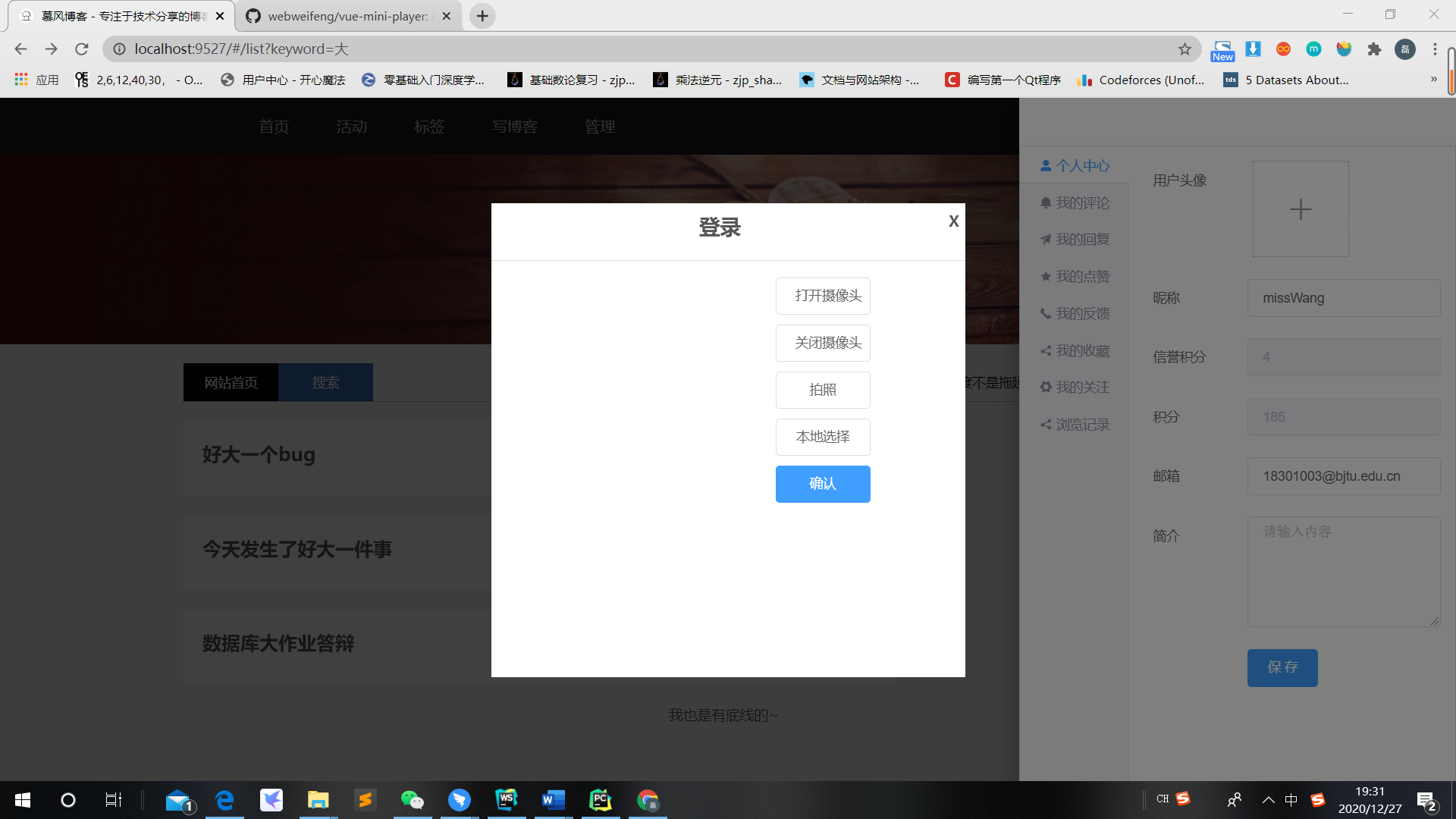
用户在浏览博客时，可以选择对相关评论者进行关注。需要注意的是自己不能关注自己，当点击关注自己的按钮时系统将会进行提示。关注成功以后将会在我的关注一栏进行显示，关注取消以后则将会在这一板块清除相关消息。

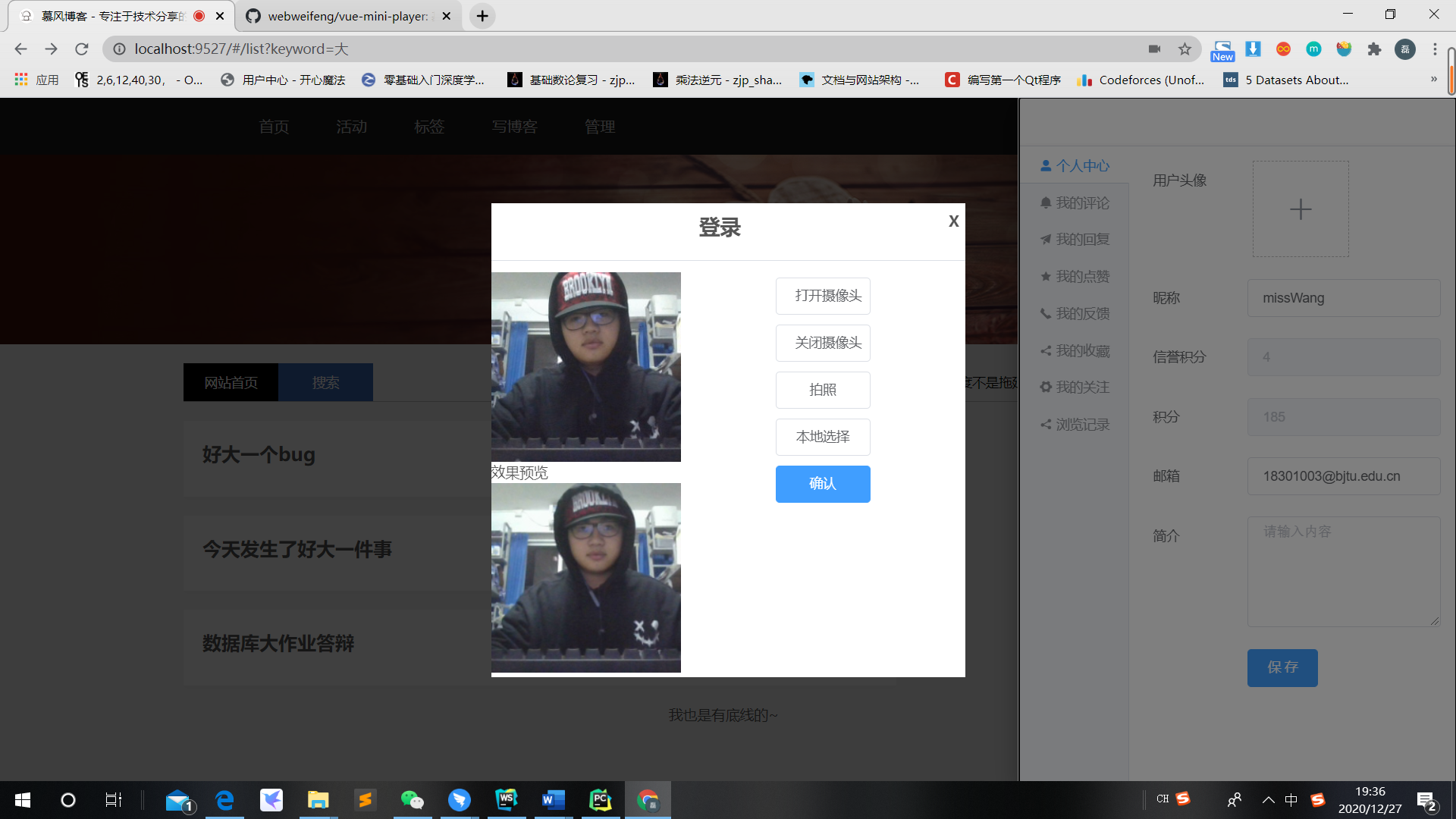
(7) 浏览记录：

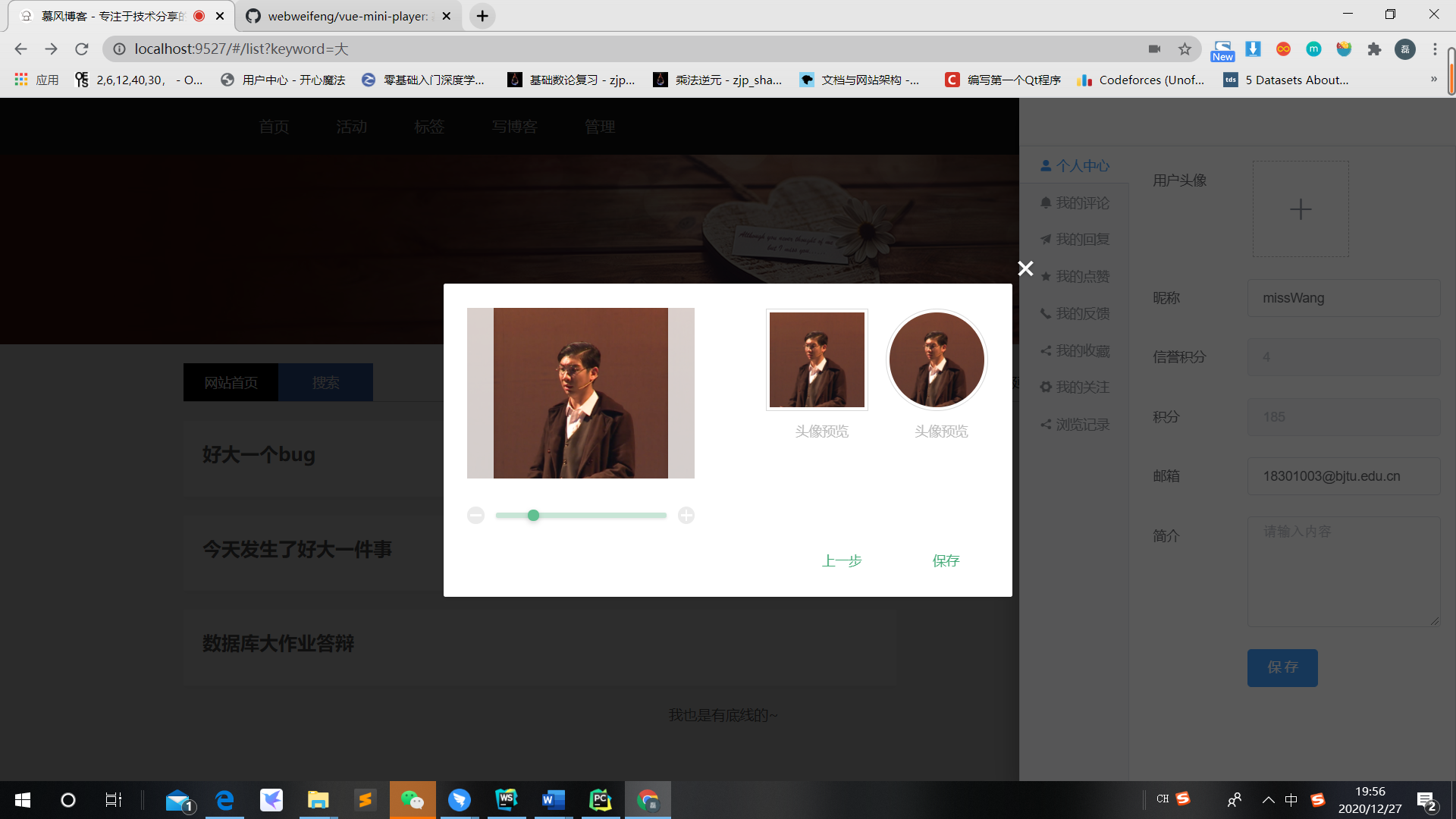
在这一栏将会记录用户的浏览记录，用户可以通过点击相关消息跳转到浏览过的博客相应界面。

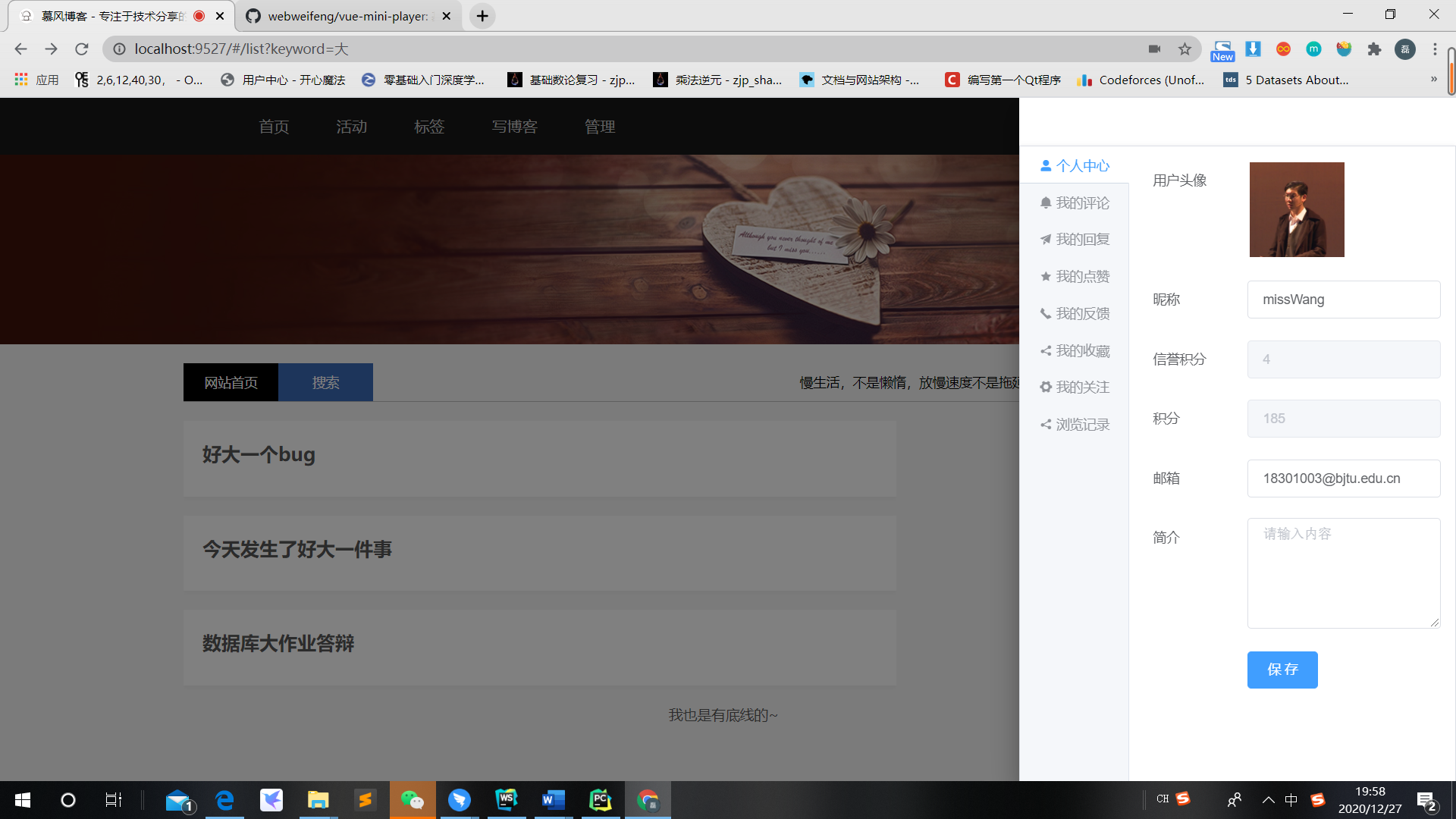
### 界面展示

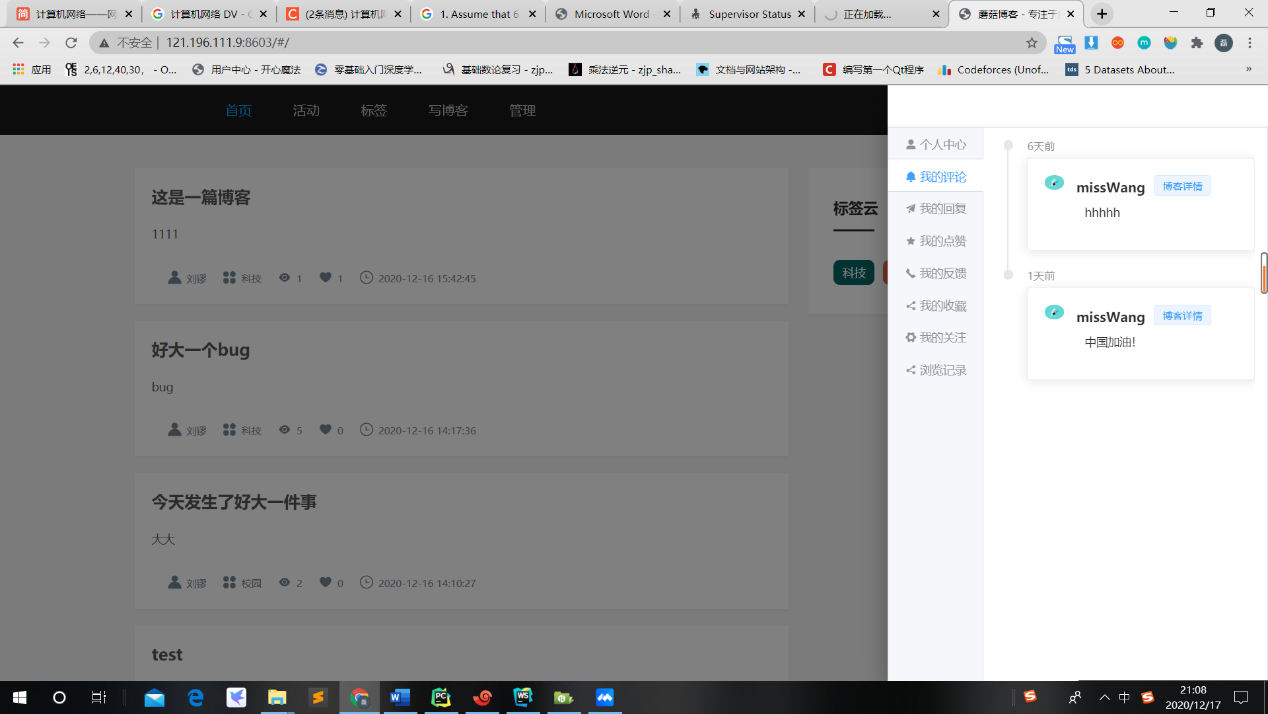












### 输入

用户操作。

### 输出

信息反馈。

### 程序逻辑

我的评论

我的回复

我的点赞

开始

我的反馈

我的收藏

我的关注

浏览记录

### 限制条件

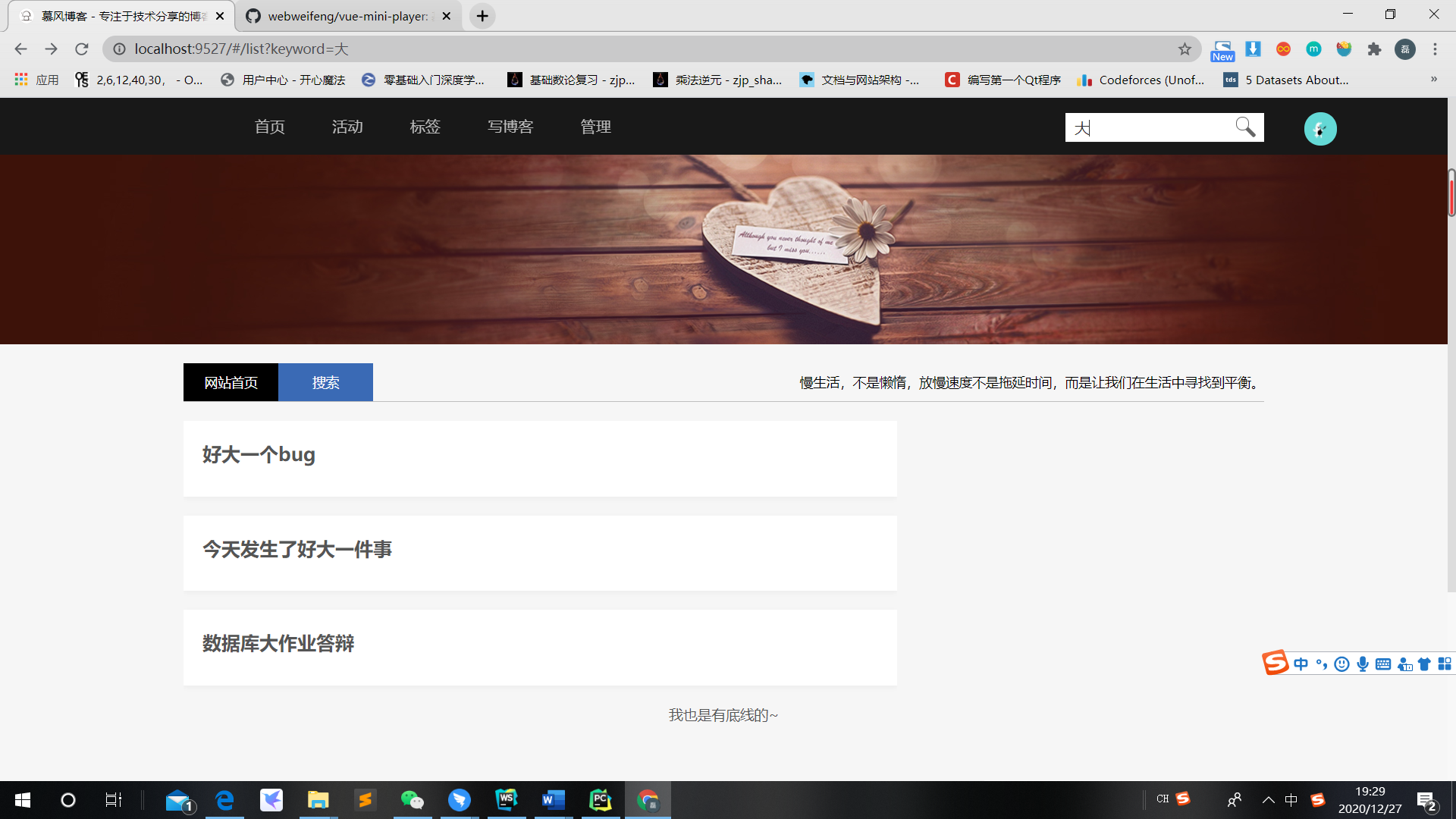
用户必须登录后才能进入个人中心。

## 搜索模块

### 功能描述

在顶部导航栏的搜索框中输入想搜索的内容，即可跳转到搜索结果页，后端将在数据库中搜索所有包含输入结果的博客并将符合搜索结果的博客返回给前端，前端将按照博客发布顺序进行展示。

### 界面展示



### 输入

需要搜索的博客内容。

### 输出

搜索结果。

### 程序逻辑

展示搜索结果

输入搜索内容

开始

### 限制条件

用户必须输入内容。

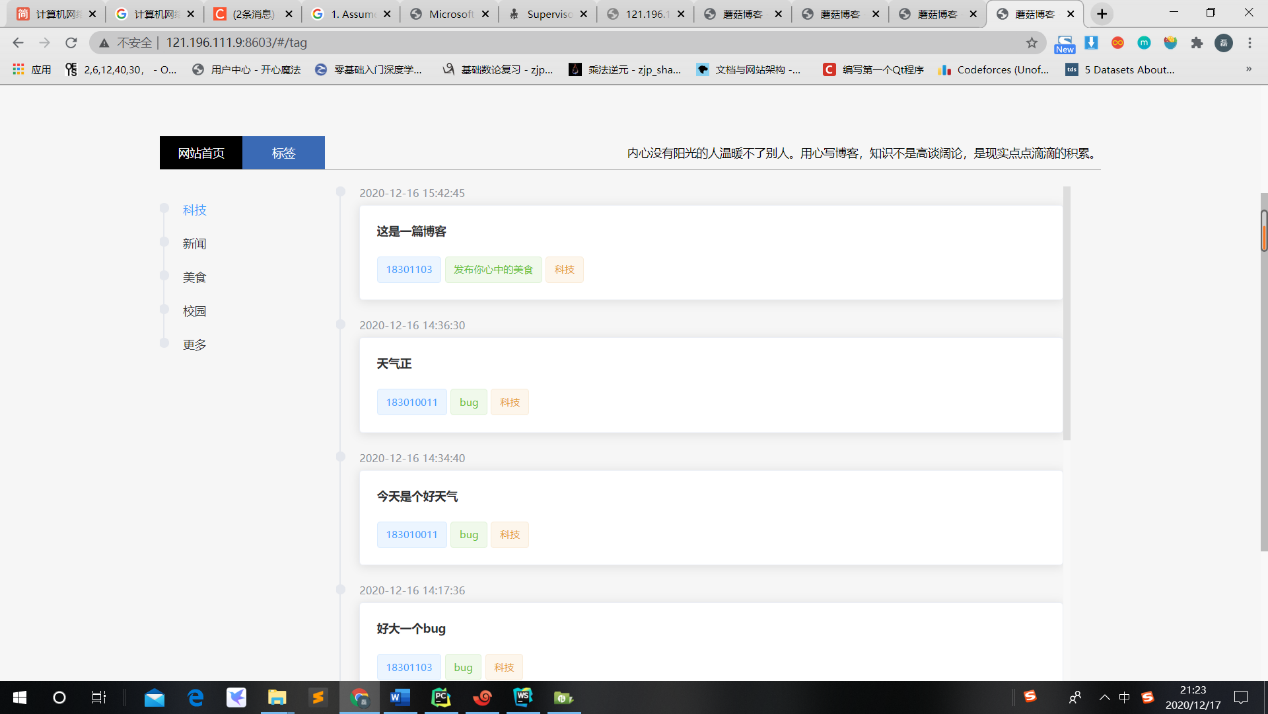
## 标签模块

### 功能描述

我们在系统中设定了“科技”，“新闻”，“美食”，“校园”，“更多”五种标签。

在标签页面中可以看到所有的标签，以及对应标签下的所有博客。

### 界面展示



### 输入

点击标签。

### 输出

标签下的博客。

### 程序逻辑

展示拥有此标签的博客

点击标签

开始

### 限制条件

无。

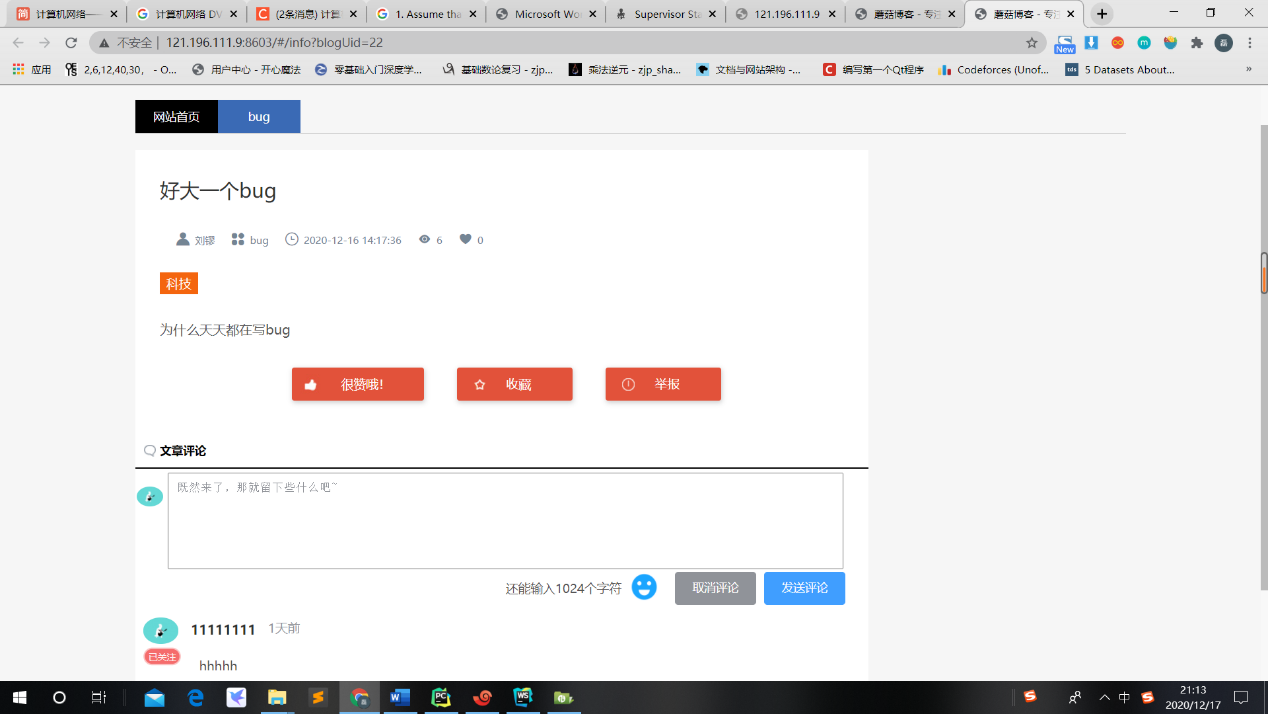
## 阅读模块

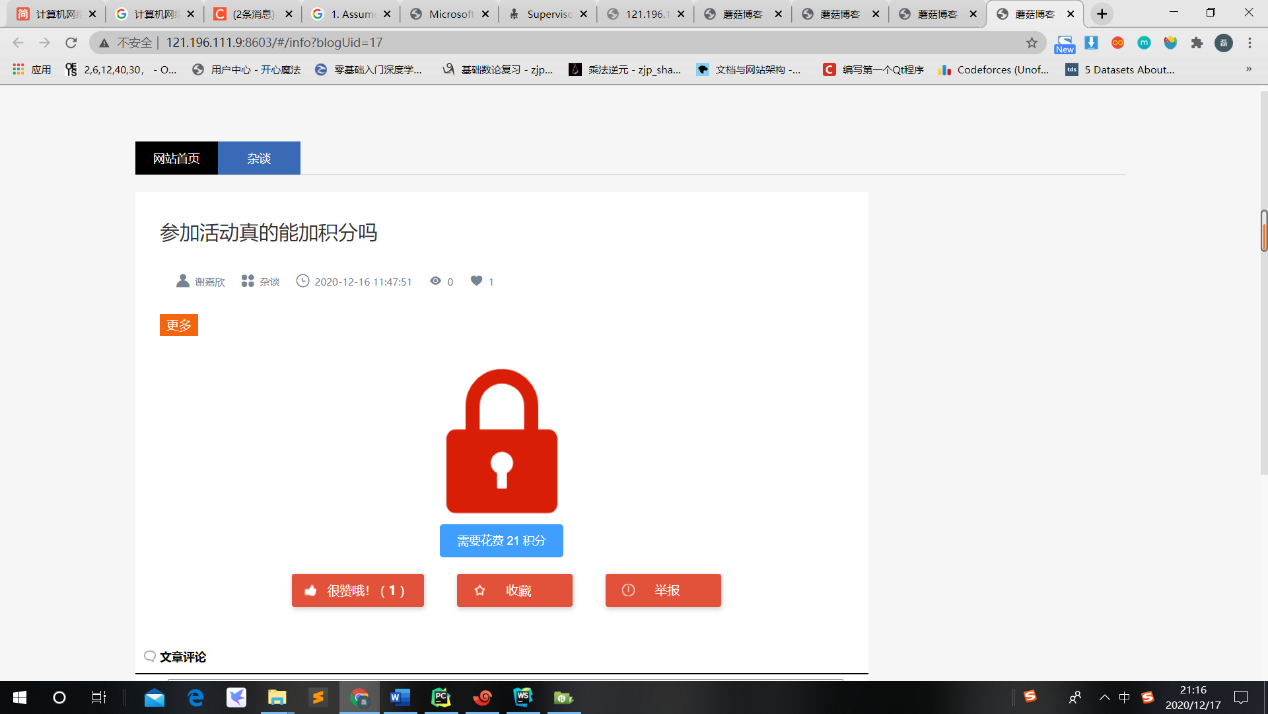
### 功能描述

点击博客标题即可进入博客阅读界面，如果该博客不需要花费积分，或者现登录用户已经支付过该博客所需积分，则可以看到博客内容。

如果该博客收费，并且用户还没有支付过所需积分，则无法看到博客内容。点击下方支付按钮，并且积分充足才能解锁并看到博客内容。

### 界面展示





### 输入

点击博客进行阅读。

### 输出

展示博客具体内容。

### 程序逻辑

查看博客

开始

查看博客

支付积分

### 限制条件

部分博客需要消耗积分。

## 审核举报模块

### 功能描述

当用户阅读文章，点击举报按钮并填写举报理由之后，成功提交，即可完成一次举报操作。

当用户举报博客后，举报信息会提交到数据库，并显示在管理员界面。

登录管理员界面，进入“管理”页，即可看到刚才用户提交的举报请求。

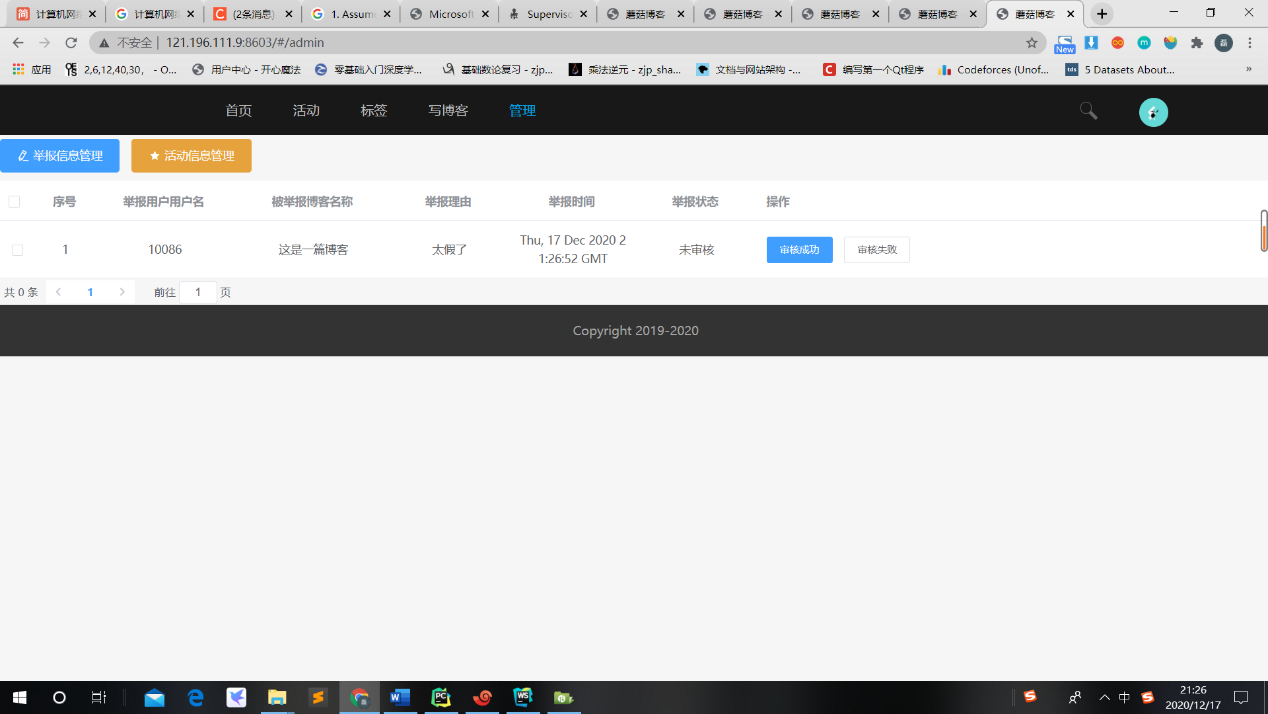
可以看到对于一条举报信息，管理员可以根据用户的举报文章和理由进行判断是否举报是否有效。

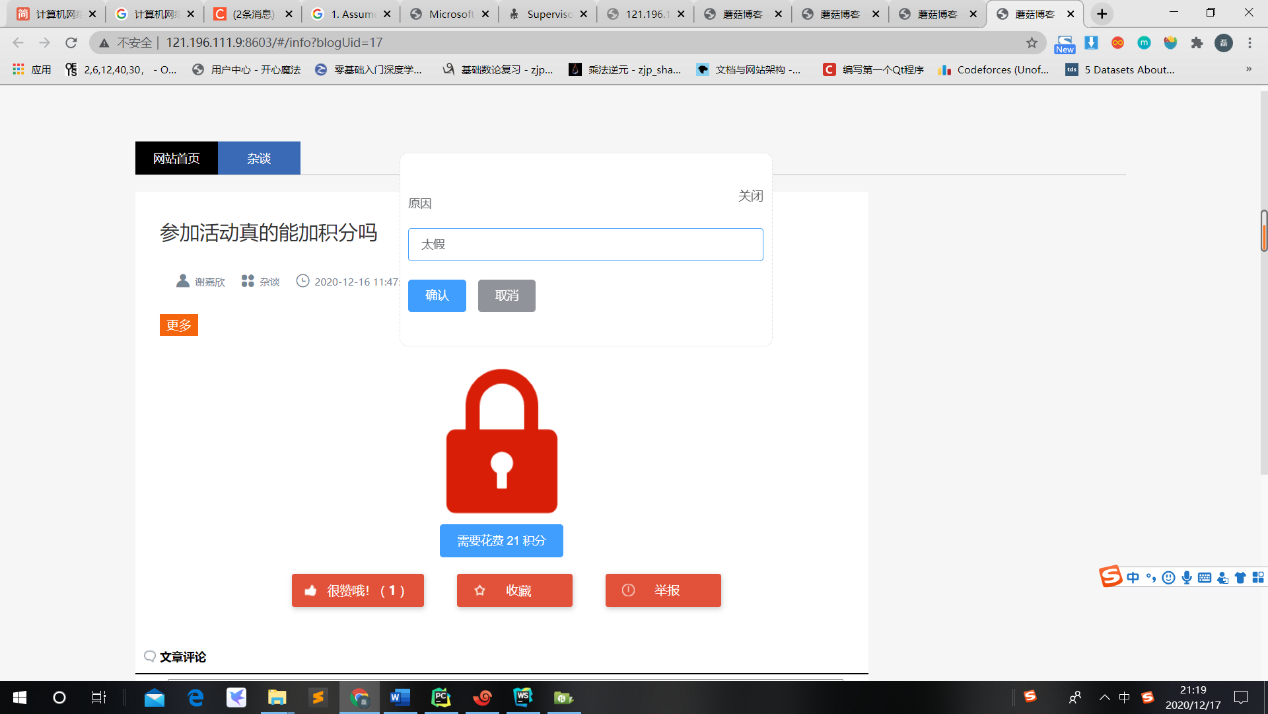
若本次举报有效，管理员点击审核成功，举报者的信誉积分未满则会增加一分，相应的被举报文章的作者会相应的减少一分。并且双方都会收到举报的结果通知。

若本次举报无效，管理员点击审核失败，则认为本次举报是恶意举报，则举报者扣除一分的信誉积分，而原文作者不受影响，也不会收到通知。

当用户信誉积分扣为0时，则会受到封号处罚。

### 界面展示





### 输入

点击博客进行举报并输入举报信息。

### 输出

对应博客进入审核举报流程。

### 程序逻辑

举报博客

开始

审核

管理员登录

### 限制条件

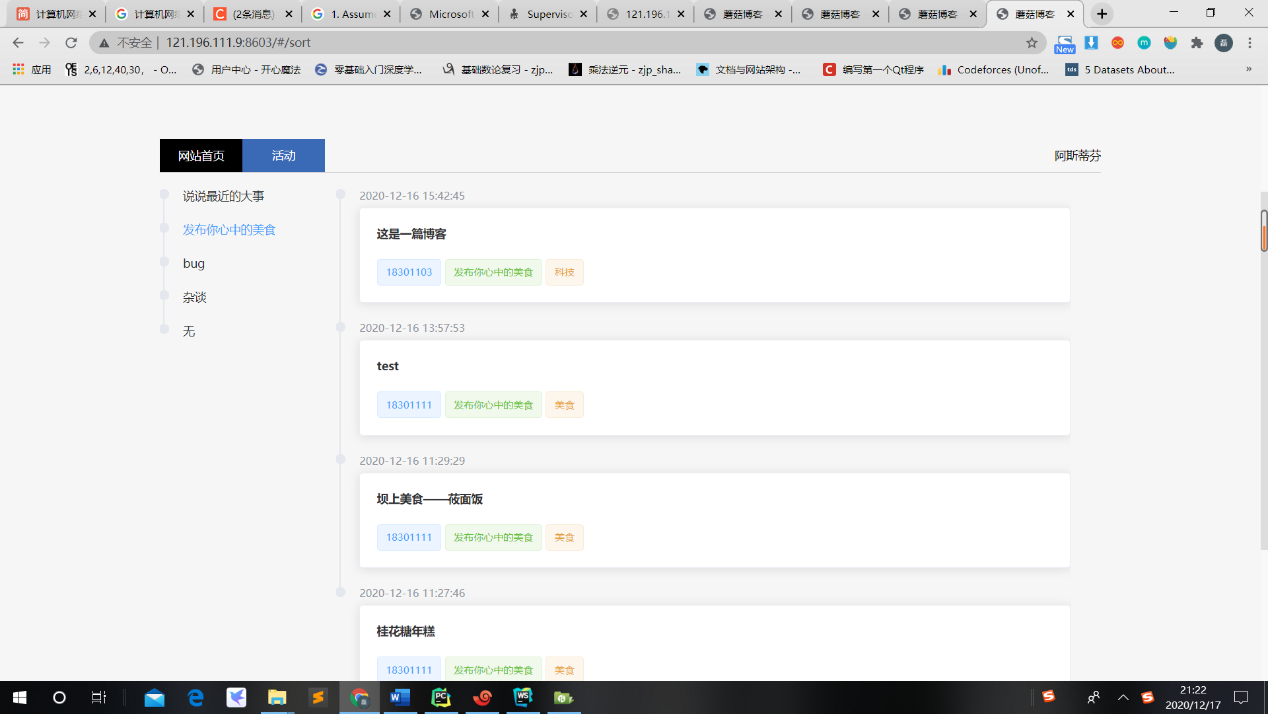
无。

## 活动模块

### 功能描述

在活动页，用户可以通过点击相关活动名称浏览参与了相关活动的所有博客，并可以进行点击跳转。博客根据时间排列。

### 界面展示



### 输入

点击活动。

### 输出

展示参加该活动的博客列表。

### 程序逻辑

展示参加此活动的博客

点击活动

开始

### 限制条件

无。

## 活动管理模块

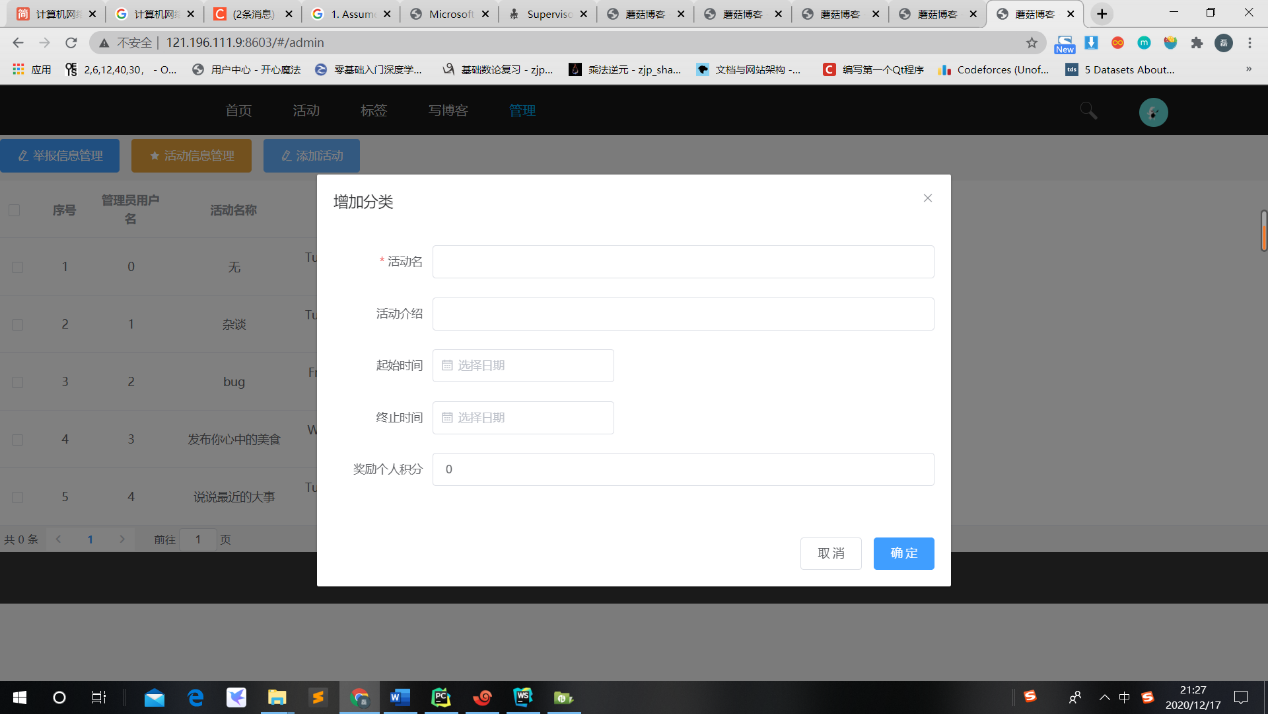
### 功能描述

管理员登陆后可以进入管理界面。在这一界面管理员可以查看历史发布的活动记录，同时可以新增活动。

在新增活动时，管理员需要填写活动的名称、描述、开始时间、结束时间以及活动奖励积分。其中开始时间和结束时间都是默认从00：00开始，开始时间最早为今天，结束时间将会万余开始时间进行设置。

### 界面展示





### 输入

输入举办活动的相关信息。

### 输出

创建新活动。

### 程序逻辑

增删改查活动

管理员登录

开始

### 限制条件

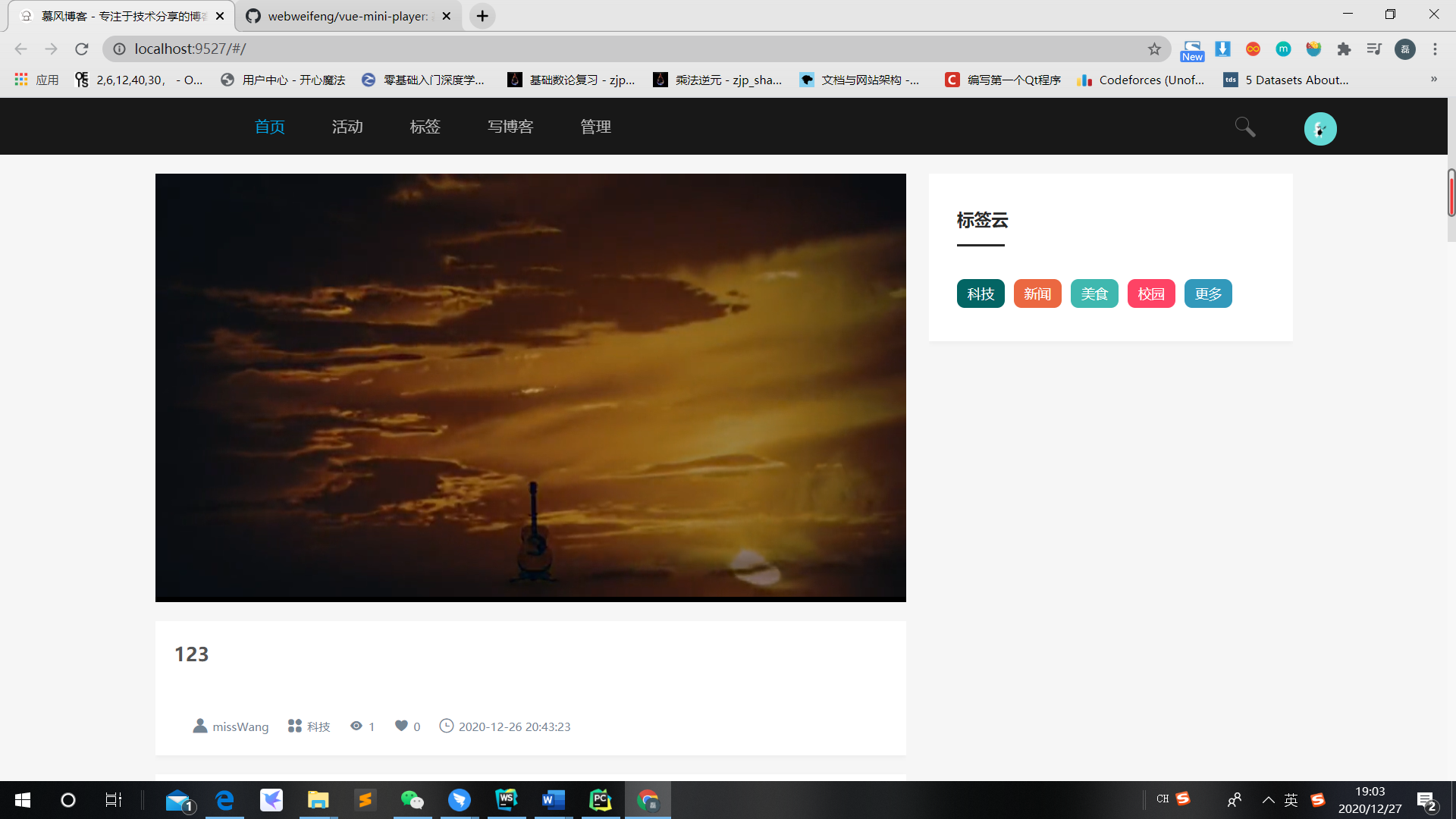
无。

## 视频播放模块

### 功能描述

在首页可以播放音视频，用户可以伴随优美的歌声浏览博客，或者观看精彩的视频，提升用户的使用体验。

### 界面展示



### 输入

打开首页。

### 输出

播放音频。

### 程序逻辑

播放音频

打开首页

开始

### 限制条件

无。

# 接口实现

## index

**getNewBlog()**

url:'/index/getNewBlog',  
method: 'get',  
params:{  
 currentPage:1,//当前页  
 pageSize:10//每页大小,即每次返回第( (currentPage-1)\*pageSize,currentPage\*pageSize)个博客，按时间排序  
}  
data:{  
 total:100,//博客总数量  
 size:10,//单页page博客数量  
 currentPage:1//现在位于第几页  
 records:[  
{  
 title: '标题',  
 author: '作者',  
 labels: ['标签','标签']   
 summary: '概括',   
 clickCount: 100,  
 likeCount: 200,  
 time: '2020-12-2'  
}，  
{  
 title: '标题',  
 author: '作者',  
 label: '标签',   
 summary: '概括',   
 clickCount: 100,  
 likeCount: 200,  
 time: '2020-12-2'  
}]  
}

## TagCloud

**getHotTag()**

url:'/index/getHotTag'  
method: 'get',  
data:[{  
 uid:'1',//数组下标  
 name：'标签名'  
}]

## info

**getBlogByUid**()

url:'/api/getBlogByUid'  
method: 'get'  
params:{  
 blog\_id:'博客id'  
}，  
data:{  
 blogSort:'',//活动名  
 title: '标题',  
 author: '作者',  
 labels: ['标签','标签']   
 summary: '概括',   
 clickCount: 100,  
 likeCount: 200,  
 time: '2020-12-2'  
}

**getWebConfig()**

url:'/index/getWebConfig'  
method: 'get'  
data:{  
 createTime:'',//现在时间  
 openAdmiration:'1',//下面两条初步默认为1  
 openComment:'1'  
}

**payCreditByUid()**

url:'/web/comment/payCreditByUid'  
method: 'get'  
params:{  
 blogUid:'博客id'  
}，  
data:{  
 code:'',  
 message:''//code是error，message为积分不足等原因  
}

**addComment()**

url:'/web/comment/add'  
method:'post'  
params:{  
 blogUid:'',  
 source:'',  
 userUid:'',  
 content:'',  
   
}

## LikeAndCollect

**praiseBlog()**

url:'/api/praiseBlogByUid'  
method:'get'  
params:{  
 uid:'博客id'  
}  
data:{  
 code:'',  
 message:'',  
 number:100//表中点赞数+1,返回+1后点赞数，一个int  
}

**getPraiseCount()**

url:'/api/getBlogPraiseCountByUid'  
method:'get'  
params:{  
 uid:'博客id'  
}  
data:{  
 100//表中点赞数  
}

**collectBlog()**

url:'/api/addCollectBlog'  
method:'get'  
params:{  
 uid:'博客id'，  
 collectName:''  
}  
data:{  
 code:''  
}

**reportBlog()**

url:'/api/reportBlog'  
method:'get'  
params:{  
 uid:'',//blogid  
 reportReason:''//举报理由  
}  
data:{  
 code:'',  
 message:''  
}

## Loginbox

**localLogin()**

url:'/login/login'  
method:'post'  
params:{  
 userName:'',  
 passWord:''  
}  
data:{  
 code:'',  
 message:''//返回信息,比如恭喜注册成功，密码不正确等  
}

**localRegister()**

url:'/login/register'  
method:'post'  
params:{  
 userName:'',  
 passWord:''，  
 email:'',  
 nickName:''  
}  
data:{  
 code:'',  
 message:''//返回信息,比如恭喜注册成功，密码不正确等  
}

## home

**getCommentListByUser()**

url:'/web/comment/getListByUser'  
method:'post'  
params:{  
 pageSize:10,  
 currentPage:1//只返回最新10个评论  
}  
data:{  
 commentList:[{  
 uid:'',  
 createTime:'',  
 user:{  
 nickName:'',  
 source:'BLOG\_INFO',//默认  
 soucreName:'博客详情'//默认,  
 content:'1'  
   
 }  
 }],  
 replyList:[{  
 uid:'',  
 createTime:'',  
 user:{  
 nickName:'',  
 source:'BLOG\_INFO',//默认  
 soucreName:'博客详情'//默认,  
 content:'1'  
   
 }   
 }],  
}

**getPraiseList()**

url:'/web/comment/getPraiseListByUser'  
method:'post'  
params:{  
 pageSize:10,  
 currentPage:1//只返回最新10个赞  
}  
data:{  
 records:[{  
 uid:'',  
 createTime:'',  
 blog:{  
 uid:'',  
 title:''  
 }  
   
 }]  
}

**authVerify()**

url: '/oauth/verify/' + params //这里传回去一个token，参数在URL里，用（/oauth/verify/<params>）当路由，可以查一下  
method:'get'  
data:{  
 uid:'',  
 nickName:'',  
 email:'',  
 credit:'',  
 role:'',  
 reputation:''  
}

**getFeedbackList()**

url:'/oauth/getFeedbackList'  
method:'get'  
data:{  
 records:[{  
 uid:'',//举报人id  
 createTime:'',//举报时间  
 title:''.//举报博客title  
 reason:'',//举报原因  
 status:'',//举报状态，0，1，2  
   
 }]  
}

**getHistoryListByUser()**

url:'/web/comment/getHistoryListByUser'  
method:'post'  
data:{  
 records:[{  
 blog\_id:'',//博客id  
 createTime:'',//时间  
 title:''.//博客title  
   
 }]  
}

**getCollectListByUser()**

url:'/web/comment/getCollectListByUser'  
method:'get'  
data:{  
 records:[{  
 blog\_id:'',//博客id  
 createTime:'',//时间  
 name:''.//博客title   
 }]  
}

**getFollowListByUser()**

url:'/web/comment/getFollowListByUser'  
method:'get'  
data:{  
 records:[{  
 follower\_id:'',//关注人id  
 createTime:'',//时间  
 nickName:''.//关注人的昵称   
 }]  
}

## blog

**getTagList()**

url:'/tag/getTagList'  
method:'get'  
data:{  
 records:[{  
 uid:1  
 content:'标签'  
 }]  
}

**getBlogSortList()**

url:'/blogSort/getList'  
method:'get'  
data:{  
 records:[{  
 uid:1  
 name:'活动名'  
 }]  
}

**addBlog()**

url:'/blog/add'  
method:'post'  
params:{  
 form: {  
 title: '',  
 summary: '',  
 content: '',  
 tagUid: '1，2',//标签id以‘，’连接的字符串  
 blogSortUid:'',//活动id  
 clickCount: 0  
 }  
}  
data:{  
 code:'success',  
 message:'发布博客成功'  
}

## FollowBtn

**getFollowedByUid()**

url:'/api/getFollowedByUid'  
method:'get'  
params:{  
 uid:'用户id'  
}  
data:{  
 liked:'false'//是否已经关注这个人，是返回true,否则返回false  
}

**followByUid()**

url:'/api/FollowByUid'  
method:'get'  
params:{  
 uid:'用户id'  
}  
data:{  
 code:'success'//本来关注的取关，本来没关注的关注  
}

## sort

**getSortList()**

url:'/sort/getSortList'  
method:'get'  
data:{  
 records:[{  
 id:'',  
 content:'',  
 description:''  
 }]  
}

**getArticleBySort()**

url:'/sort/getArticleBySort'  
method:'get'  
params:{  
 id:'活动id'  
}  
data:{  
 records:[{  
 title: '标题',  
 author: '作者',  
 labels: [{uid:'',content:''}],  
 blogSort:{uid:'',sortName:''}  
 time: '2020-12-2'  
 }]  
}

## tag

**getTagList()**

url:'/tag/getTagList'  
method:'get'  
data:{  
 records:[{  
 id:'',  
 name:''  
 }]  
}

**getArticleByTagUid()**

url:'/tag/getArticleByTagUid'  
method:'get'  
params:{  
 tagUid:''  
}  
data:{  
 records:[{  
 title: '标题',  
 author: '作者',  
 labels: [{uid:'',content:''}],  
 blogSort:{uid:'',sortName:''}  
 time: '2020-12-2'  
 }]  
}

## admin

**getReportList()**

url:'/admin/getReportList'  
method:'get'  
data:{  
 records:[{  
 uid: '1',  
 blog\_id: '1',  
 title: 'test',  
 reason: 'ghs',  
 time: '2020-12-7',  
 status: '0'  
 }]  
}

**getSortList()**

url:'/admin/getSortList'  
method:'get'  
data:{  
 records:[{  
 uid: '1',  
 sortName: 'test',  
 startTime: '2020-12-7',  
 endTime: '2020-12-7',  
 credit: 10   
 }]  
}

**addBlogSort()**

url:'/admin/blogSort/add'  
method:'post'  
params:{  
 sortName: '',  
 description: '',  
 startTime: '',  
 endTime: '',  
 credit: 0  
}  
data:{  
 message:'添加活动成功'  
}

**workReport()**

url:'/admin/workReport'  
method:'post'  
params:{  
 uid:''，//用户名  
 title:'',  
 reason:'',  
 time:'',  
 status:'',  
 blog\_id:''  
}  
data:{  
 code:'',  
 message:''  
}

## CommentList

**getCommentList()**

url:'/web/comment/getList'  
method:'post'  
params:{  
 pageSize:10,  
 currentPage:0,//只返回最新10个评论  
 source:'',  
 blogUid:''  
}  
data:{  
 records:[{  
 uid:'',  
 createTime:'',  
 user:{  
 id:'',//发表该评论用户的id  
 nickName:'',  
   
   
 },  
 content:'1'  
 }]  
}